LIBRARY MANAGEMENT SYSTEM

Higher National Diploma in Software Engineering

DBMS-2 Project Documentation

24.1F

National Institute of Business Management

Kandy Regional Center

No 2, Asgiriya Road,

Kandy

LIBRARY MANAGEMENT SYSTEM

Higher National Diploma in Software Engineering

DBMS-2 Project Documentation

24.1F

M.M.M AMRY	- KAHDSE24.1F - 023
A. DHANUSHANANDAN	- KAHDSE24.1F - 028
M.A.M AMMAR	- KAHDSE24.1F – 026
M.Z.F.ZEENA	- KAHDSE24.1F - 022

The project is submitted in partial fulfilment of the requirement of the Higher National Diploma of Software Engineering of National Institute of Business Management.

November 2024

DECLARATION

We declare this report was not a copy of a document done by any organization, university or any other institute and was not copied from the internet or other sources. This document is proprietary and exclusive property of the following mentioned group. No part of this document in whole or in part, may be reproduced, stored, transmitted, or used for design purposes without the prior written permission of NIBM. This report is a unique document and all the members actively participated in its accomplishment of it.

REGISTER NUMBER	**Name**	**Signature**
KAHDSE24.1F-028	A.DHANUSHANANDAN	
KAHDSE24.1F-023	M.M.M AMRY	
KAHDSE24.1F-026	M.A.M AMMAR	
KAHDSE24.1F-022	M.Z.F.ZEENA	

Certified by:

Lecturer : Mr. Manjula kulathunga.

Date of submission : 01/11/2024

Signature :

Contents

INTRODUCTION

This Project aims to create a Oracle Database using PL/SQL for Library management system. In our project we include CRUD operations for the functions Like Books, Member and Borrowed. Additionally, we use Exception handling, restrict permissions for users using user management and created reports for maintenance.

INTRODUCTIONS TO BUSINESS

We develop an Oracle Database using PL/SQL for the Library management system to manage Books, Members details, and track Borrowed details. Our system helps to easily retrieve or add new Books, Members or Borrowed.

REASON TO USE ORACLE DATABASE

It's a leading Relational database management system for its high performance and capabilities. It's a scalable platform so it helps to manage data incentive applications.

- High Availability : It's a continuous work database through oracle data guard. If an unexpected failure occurs its use a secondary database.
- Backup : Its use Recovery Manager to recover during online and archived backups.
- Scalability : RAC allows to run multiple instances on different servers its help to improve the performance and availability.
- Multitenant Architecture: Oracle use Multitenant architecture to simplifies the management.
- Security : Oracle provide TDE and Oracle database Vault to protect sensitive information from unauthorized access.
- Data redaction : Its help to mask the sensitive information helps to enhance security.
- PL/SQL Support : facilitate complex operations and data manipulation.

NEED FOR AN ORACLE DATBASE FOR LMS

Developing a LMS with oracle database system is significantly improve the efficiency, scalability and security.

- Scalability and Performance : Its work with high load so its suitable when students and lectures can use simultaneously during an exam period and reading online time.
- Automatic Scalability : Its Automatically scale resources up or down. Its helps user to loads without delay or performance degressions.
- Security : Oracle provides advance security measures and data encryptions its help to secure sensitive information. Its helps to improve user privacy and securing the library resources.
- Data Recover : In case of hardware failure or data corruption oracle provide backup resources to recover the data. It improves the LMS trust.
- Automation and Efficiency : Minimize the manual database management. Use to auto scaling and auto tuning. So, Library staffs can focus on the users without focus on the technical maintenance.
- Flexibility : libraries can enhance their service delivery, ensuring they meet the evolving needs of their users efficiently and effectively.

REQUIREMENTS OF ANALYSIS

FUNCTIONAL REQUIREMENTS

- Manage Books details like Add new Books, Update Books, Delete Books, and View Books.
- Manage Members details like Add new member, Update member, Delete member, view Members and their user roles.
- Track the Borrowed books and returned books.

NON-FUNCTIONAL REQUIREMENTS

- Ensure the data security.
- Scalable database.

DATABASE DESIGN

LOGICAL DESIGN

Entities : Book, Member, Borrowed.

Relationship : One to Many Relationship (Member - Borrowed , Book - Borrowed).

PHYSICAL DESIGN

Database have included Book, Member, Borrowed tables.

1.Book Table – Store the Book details.

Column name	**Description**
Book_Id	Varchar(100) / Primary key / Check (Book_Id LIKE 'B-%') not null
Book_Title	Varchar(100)
Book_Author	Varchar(100)
Book_Add_Date	Date
Book_Copies	int

2.Member Table – Store the Member details.

Column name	Description
Member_Id	Varchar(100) / Primary key / Check (Member_Id LIKE 'M-%') not null
Member_Name	Varchar(100)
Member_Phone	Varchar(10) / used because can be use +94
Member_add_date	Date
Member_Role	Varchar(100) / Check (Member_Role IN ('admin','user'))

3.Borrwed Table – Store the Borrowed and Return details.

Column name	Description
Borrowed_Id	Varchar(100) / Primary key / not null
Member_Id	Varchar(100) / references Member(Member_Id)
Book_Id	Varchar(100) / references Book(Book_Id)
Borrowed_date	Date
Return_date	Date
Book_Returned	CHAR(1) default 'N'

DATABASE TABLE CREATE CODE

```
--Book Details
create TABLE Book(
    Book_Id VARCHAR(100) primary KEY CHECK (Book_Id LIKE 'B-%') not null,
    Book_Title VARCHAR(100),
    Book_Author VARCHAR(100),
    Book_add_date DATE,
    Book_copies int

);
--Member Details
create TABLE Member(
```

```
    Member_Id VARCHAR(100) primary KEY CHECK (Member_Id LIKE 'M-%') not null,
    Member_Name VARCHAR(100),
    Member_Phone VARCHAR(10),
    Member_add_date DATE,
    Member_Role VARCHAR(100) CHECK (Member_Role IN ('admin','user'))

);

--Borrowed Book Details
create TABLE Borrowed(
    Borrowed_Id VARCHAR(100) primary KEY not null,
    Member_Id VARCHAR(100) references Member(Member_Id) ON DELETE CASCADE,
    Book_Id VARCHAR(100) references Book(Book_Id) ON DELETE CASCADE,
    Borrowed_date DATE,
    Return_date DATE,
    Book_Returned CHAR(1) default 'N'
);
```

CRUD OPERATIONS FOR THE LMS

1.BOOK

INSERT PL/SQL PROCEDURE CODE

```
--Insert a Book (Procedure)
Create OR REPLACE PROCEDURE insertBook(B_Id VARCHAR,B_Title VARCHAR,B_Author
VARCHAR,B_add_date DATE,B_copies int)
IS
    cursor c_book IS SELECT Book_Id from BOOK;
    book_id VARCHAR(100);
    B_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_book;

        DBMS_OUTPUT.PUT_LINE('opened cursor');
    LOOP

        DBMS_OUTPUT.PUT_LINE('inside loop');
        FETCH c_book into book_id;

        DBMS_OUTPUT.PUT_LINE('fetching');
        EXIT WHEN c_book%NOTFOUND;
        IF book_id = B_Id THEN
            -- DBMS_OUTPUT.PUT_LINE('Book ID Already Exists Try New');
```

```
        DBMS_OUTPUT.PUT_LINE('inside if');
            B_exsist := TRUE;
            EXIT;

        DBMS_OUTPUT.PUT_LINE('exit if');
        ELSE
            B_exsist := FALSE;
        END IF;
    END LOOP;

        DBMS_OUTPUT.PUT_LINE('exit from if and loop');
    close c_book;

        DBMS_OUTPUT.PUT_LINE('cursor closed');

    IF B_exsist = TRUE THEN
        DBMS_OUTPUT.PUT_LINE('inside 2nd if');

        DBMS_OUTPUT.PUT_LINE('Book Id already Exsist Try New!');
    ELSE

        DBMS_OUTPUT.PUT_LINE('inside 2nd if else');
        INSERT into Book(Book_Id,Book_Title,Book_Author,Book_add_date,Book_copies)
values(B_Id,B_Title,B_Author,B_add_date,B_copies);
         DBMS_OUTPUT.PUT_LINE('Book Inserted complete');

    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-Insert the values using execute keyword hardcode values
EXECUTE INSERTBOOK('B-001','POC','CAPJ',DATE'2024-10-23',10);
EXECUTE INSERTBOOK('B-002','POC','CAPJ',DATE'2024-10-20',5);
EXECUTE INSERTBOOK('B-003','POC','CAPJ',DATE'2024-10-20',1);
```

Method = 2 using Prompt

```
--M2-Insert values using prompt
SET SERVEROUTPUT ON
ACCEPT book_id char PROMPT 'Enter Book ID: Start B-'
ACCEPT book_title char PROMPT 'Enter Book Title:'
ACCEPT book_author char PROMPT 'Enter Book Author:'
ACCEPT book_date DATE PROMPT 'Enter today date: YYYY-MM-DD'
ACCEPT book_copies number PROMPT 'Enter No Books Copies : '
DECLARE
    bookid Book.BOOK_ID%TYPE;
    booktitle Book.Book_Title%TYPE;
    bookauthor Book.Book_Author%TYPE;
    bookadddate Book.Book_add_date%TYPE;
    bookcopies Book.Book_copies%TYPE;
BEGIN
    bookid :='&book_id';
    booktitle :='&book_title';
    bookauthor :='&book_author';
    bookadddate := TO_DATE('&book_date','YYYY-MM-DD');
    bookcopies := '&book_copies';
        DBMS_OUTPUT.PUT_LINE('inputs collected');
    INSERTBOOK(bookid,booktitle,bookauthor,bookadddate,bookcopies);
END;
```

UPDATE PROCEDURE

```
--Update a Book (Procedure)
Create OR REPLACE PROCEDURE updateBook(B_Id VARCHAR,B_Title VARCHAR,B_Author
VARCHAR,B_add_date DATE,B_copies int)
IS
    cursor c_book IS SELECT Book_Id from BOOK;
    book_id VARCHAR(100);
    B_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_book;
    LOOP
        FETCH c_book into book_id;
        EXIT WHEN c_book%NOTFOUND;
        IF book_id = B_Id THEN
            B_exsist := TRUE;
            EXIT;
        ELSE
            B_exsist := FALSE;
```

```
        END IF;
    END LOOP;
    close c_book;

    IF B_exsist = TRUE THEN
        Update Book SET
Book_Title=B_Title,Book_Author=B_Author,Book_add_date=B_add_date,Book_copies=B_copies
where Book_Id=B_Id;
         DBMS_OUTPUT.PUT_LINE('Book Update complete');
    ELSE
         DBMS_OUTPUT.PUT_LINE('Book ID NOT Exists Try New');
    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-Update the values using execute keyword hardcord values
EXECUTE updateBook('B-002','POC-2','CAPJ',DATE'2024-10-25',5);
EXECUTE updateBook('B-003','POC-2','CAPJ',DATE'2024-10-25',9);
```

Method = 2 using Prompt

```
--M2-Update values using prompt
SET SERVEROUTPUT ON
ACCEPT book_id char PROMPT 'Enter Exist Book ID: Start B-'
ACCEPT book_title char PROMPT 'Enter Book Title:'
ACCEPT book_author char PROMPT 'Enter Book Author:'
ACCEPT book_date DATE PROMPT 'Enter today date: YYYY-MM-DD'
ACCEPT book_copies number PROMPT 'Enter Book Copies:'
DECLARE
    bookid Book.BOOK_ID%TYPE;
    booktitle Book.Book_Title%TYPE;
    bookauthor Book.Book_Author%TYPE;
    bookadddate Book.Book_add_date%TYPE;
    bookcopies Book.BOOK_COPIES%TYPE;
BEGIN
    bookid :='&book_id';
    booktitle :='&book_title';
```

```
    bookauthor :='&book_author';
    bookadddate := TO_DATE('&book_date','YYYY-MM-DD');
    bookcopies :='&book_copies';
    updateBook(bookid,booktitle,bookauthor,bookadddate,bookcopies);
END;
```

VIEW PROCEDURE

VIEW ALL

```
--view all books
CREATE OR REPLACE PROCEDURE ViewAllBook
IS
    cursor c_book IS
        SELECT Book_Id, BOOK_TITLE, BOOK_AUTHOR, BOOK_ADD_DATE,BOOK_COPIES
        FROM BOOK;
    book_id         VARCHAR(100);
    book_title      VARCHAR(100);
    book_author     VARCHAR(100);
    book_add_date   DATE;
    book_copies number;
BEGIN
    OPEN c_book;
    LOOP
        FETCH c_book INTO book_id, book_title, book_author, book_add_date ,
book_copies;
        EXIT WHEN c_book%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Book Information:');
        DBMS_OUTPUT.PUT_LINE('Book_Id: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book_Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book_Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Book_Add_Date: ' || TO_CHAR(book_add_date, 'YYYY-MM-
DD'));
        DBMS_OUTPUT.PUT_LINE('Book_Copies: ' || book_copies);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_book;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--dispaly all the books
EXECUTE VIEWALLBOOK;
```

VIEW USING ID SPECIFIC

```
--Display a Book with id/title/author (Procedure)
Create OR REPLACE PROCEDURE ViewSinlgeBook(B_Id VARCHAR)
IS
    cursor c_book IS SELECT Book_Id, BOOK_TITLE, BOOK_AUTHOR, BOOK_ADD_DATE
,BOOK_COPIES FROM BOOK;
    book_id VARCHAR(100);
    book_title     VARCHAR(100);
    book_author    VARCHAR(100);
    book_add_date  DATE;
    book_copies number;
    B_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_book;
    LOOP
        FETCH c_book INTO book_id, book_title, book_author, book_add_date, book_copies;
        EXIT WHEN c_book%NOTFOUND;
        IF book_id = B_Id OR Book_Title = B_Id OR Book_author = B_Id THEN
            B_exsist := TRUE;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Book Information:');
            DBMS_OUTPUT.PUT_LINE('Book_Id: ' || book_id);
            DBMS_OUTPUT.PUT_LINE('Book_Title: ' || book_title);
            DBMS_OUTPUT.PUT_LINE('Book_Author: ' || book_author);
            DBMS_OUTPUT.PUT_LINE('Book_Add_Date: ' || TO_CHAR(book_add_date, 'YYYY-MM-
DD'));
            DBMS_OUTPUT.PUT_LINE('Book_Copies: ' || book_copies);
            DBMS_OUTPUT.PUT_LINE(' ');
            -- EXIT;
        ELSE
            B_exsist := FALSE;
        END IF;
    END LOOP;
    close c_book;

    IF B_exsist = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Book ID Not Exists Try New');
```

```
    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE ViewSinlgeBook('B-002');
EXECUTE ViewSinlgeBook('CAPJ');
EXECUTE ViewSinlgeBook('POC-2');
```

DELETE PROCEDURE

```
--Delete a Book with id (Procedure)
Create OR REPLACE PROCEDURE DeleteBook(B_Id VARCHAR)
IS
    cursor c_book IS SELECT Book_Id, BOOK_TITLE, BOOK_AUTHOR, BOOK_ADD_DATE
,BOOK_COPIES FROM BOOK;
    book_id VARCHAR(100);
    book_title     VARCHAR(100);
    book_author    VARCHAR(100);
    book_add_date  DATE;
    book_copies number;
    B_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_book;
    LOOP
        FETCH c_book INTO book_id, book_title, book_author, book_add_date ,book_copies;
        EXIT WHEN c_book%NOTFOUND;
        IF book_id = B_Id THEN
            B_exsist := TRUE;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Book Information:');
            DBMS_OUTPUT.PUT_LINE('Book_Id: ' || book_id);
            DBMS_OUTPUT.PUT_LINE('Book_Title: ' || book_title);
            DBMS_OUTPUT.PUT_LINE('Book_Author: ' || book_author);
```

```
            DBMS_OUTPUT.PUT_LINE('Book_Add_Date: ' || TO_CHAR(book_add_date, 'YYYY-MM-
DD'));
            DBMS_OUTPUT.PUT_LINE('Book_copies: ' || book_copies);
            Delete from Book where Book_Id = B_Id;
            DBMS_OUTPUT.PUT_LINE('Book Deleted');
            EXIT;
        ELSE
            B_exsist := FALSE;
        END IF;
    END LOOP;
    close c_book;

    IF B_exsist = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Book ID Not Exists Try New');
    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE DeleteBook('B-002');
```

2.MEMBER

INSERT PL/SQL PROCEDURE CODE

```
--Insert a Member (Procedure)
Create OR REPLACE PROCEDURE insertMember(M_Id VARCHAR,M_name VARCHAR,M_phone
VARCHAR,M_add_date DATE,M_role VARCHAR)
IS
   cursor c_member IS SELECT Member_Id from Member;
   member_id VARCHAR(100);
```

```
    M_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_member;
    LOOP
        FETCH c_member into member_id;
        EXIT WHEN c_member%NOTFOUND;
        IF member_id = M_Id THEN
            M_exsist := TRUE;
            EXIT;
        ELSE
            M_exsist := FALSE;
        END IF;
    END LOOP;
    close c_member;

    IF M_exsist = TRUE THEN
        DBMS_OUTPUT.PUT_LINE('Member ID Already Exists Try New');
    ELSE
        INSERT into
MEMBER(MEMBER_ID,MEMBER_NAME,MEMBER_PHONE,MEMBER_ADD_DATE,MEMBER_ROLE)
values(M_Id,M_name,M_phone,M_add_date,M_role);
         DBMS_OUTPUT.PUT_LINE('Member Inserted complete');
    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-Insert the values using execute keyword hardcord values
EXECUTE insertMember('M-001','CAPJ','0712051203',DATE'2024-10-23','admin');
```

Method = 2 using Prompt

```
--M2-Insert values using prompt
SET SERVEROUTPUT ON
ACCEPT member_id char PROMPT 'Enter Member ID: Start M-'
ACCEPT member_name char PROMPT 'Enter Member Name:'
ACCEPT member_phone char PROMPT 'Enter Member Phone:'
ACCEPT member_date DATE PROMPT 'Enter today date: YYYY-MM-DD'
ACCEPT member_role char PROMPT 'Enter Role: admin or user'
DECLARE
   memberid Member.MEMBER_ID%TYPE;
   membername Member.MEMBER_NAME%TYPE;
   memberphone Member.MEMBER_PHONE%TYPE;
   memberdate Member.MEMBER_ADD_DATE%TYPE;
   memberrole Member.MEMBER_ROLE%TYPE;
BEGIN
   memberid :='&member_id';
   membername :='&member_name';
   memberphone :='&member_phone';
   memberdate := TO_DATE('&member_date','YYYY-MM-DD');
   memberrole :='&member_role';
   INSERTMEMBER(memberid,membername,memberphone,memberdate,memberrole);
END;
```

UPDATE PROCEDURE

```
--update a Member (Procedure)
Create OR REPLACE PROCEDURE UpdateMember(M_Id VARCHAR,M_name VARCHAR,M_phone
VARCHAR,M_add_date DATE,M_role VARCHAR)
IS
   cursor c_member IS SELECT Member_Id from Member;
   member_id VARCHAR(100);
   M_exsist BOOLEAN := FALSE;
BEGIN
   OPEN c_member;
```

```
   LOOP
      FETCH c_member into member_id;
      EXIT WHEN c_member%NOTFOUND;
      IF member_id = M_Id THEN
         M_exsist := TRUE;
         EXIT;
      ELSE
         M_exsist := FALSE;
      END IF;
   END LOOP;
   close c_member;

   IF M_exsist = TRUE THEN
        UPDATE Member set
MEMBER_NAME=M_name,MEMBER_PHONE=M_phone,MEMBER_ADD_DATE=M_add_date,MEMBER_ROL
E=M_role where MEMBER_ID=M_Id;
       DBMS_OUTPUT.PUT_LINE('Member Update complete');
   ELSE
       DBMS_OUTPUT.PUT_LINE('Member ID NOT Exists Try New');
   END IF;

   EXCEPTION
   WHEN no_data_found THEN
      DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-update the values using execute keyword hardcode values
EXECUTE UpdateMember('M-001','CAPJP','0712051203',DATE'2024-10-20','admin');
EXECUTE UpdateMember('M-002','CAPJ','0712051203',DATE'2024-10-23','admin');
```

Method = 2 using Prompt

```
--M2-update values using prompt
SET SERVEROUTPUT ON
ACCEPT member_id char PROMPT 'Enter Member ID: Start M-'
ACCEPT member_name char PROMPT 'Enter Member Name:'
ACCEPT member_phone char PROMPT 'Enter Member Phone:'
ACCEPT member_date DATE PROMPT 'Enter today date: YYYY-MM-DD'
ACCEPT member_role char PROMPT 'Enter Role: admin or user'
DECLARE
   memberid Member.MEMBER_ID%TYPE;
   membername Member.MEMBER_NAME%TYPE;
   memberphone Member.MEMBER_PHONE%TYPE;
   memberdate Member.MEMBER_ADD_DATE%TYPE;
   memberrole Member.MEMBER_ROLE%TYPE;
BEGIN
   memberid :='&member_id';
   membername :='&member_name';
   memberphone :='&member_phone';
   memberdate := TO_DATE('&member_date','YYYY-MM-DD');
   memberrole :='&member_role';
   UpdateMember(memberid,membername,memberphone,memberdate,memberrole);
END;
```

VIEW PROCEDURE

VIEW ALL

```
--View all Member (Procedure)
Create OR REPLACE PROCEDURE ViewAllMembers
IS
   cursor c_member IS SELECT
Member_Id,MEMBER_NAME,MEMBER_PHONE,MEMBER_ADD_DATE,MEMBER_ROLE from Member;
   member_id VARCHAR(100);
   member_name VARCHAR(100);
   member_phone VARCHAR(10);
   member_date DATE;
   member_role VARCHAR(100);
```

```
   M_exsist BOOLEAN := FALSE;
BEGIN
   OPEN c_member;
   LOOP
      FETCH c_member into member_id,member_name,member_phone,member_date,member_role;
      EXIT WHEN c_member%NOTFOUND;

         DBMS_OUTPUT.PUT_LINE(' ');
         DBMS_OUTPUT.PUT_LINE('Member Information:');
         DBMS_OUTPUT.PUT_LINE('Member_Id: ' || member_id);
         DBMS_OUTPUT.PUT_LINE('Member_Name: ' || member_name);
         DBMS_OUTPUT.PUT_LINE('Member_phone: ' || member_phone);
         DBMS_OUTPUT.PUT_LINE('Member_Add_Date: ' || TO_CHAR(member_date, 'YYYY-MM-DD'));
         DBMS_OUTPUT.PUT_LINE('Member_Role: ' || member_role);
         DBMS_OUTPUT.PUT_LINE(' ');
         EXIT;

   END LOOP;
   close c_member;

   EXCEPTION
   WHEN no_data_found THEN
      DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE VIEWALLMEMBERS;
```

VIEW USING ID SPECIFIC

```
--View single Member ID/name (Procedure)
```

```
Create OR REPLACE PROCEDURE ViewsingleMembers(M_Id VARCHAR)
IS
    cursor c_member IS SELECT
Member_Id,MEMBER_NAME,MEMBER_PHONE,MEMBER_ADD_DATE,MEMBER_ROLE from Member;
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(10);
    member_date DATE;
    member_role VARCHAR(100);
    M_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_member;
    LOOP
        FETCH c_member into member_id,member_name,member_phone,member_date,member_role;
        EXIT WHEN c_member%NOTFOUND;
        IF member_id = M_Id OR MEMBER_NAME = M_Id THEN
            M_exsist := TRUE;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Member Information:');
            DBMS_OUTPUT.PUT_LINE('Member_Id: ' || member_id);
            DBMS_OUTPUT.PUT_LINE('Member_Name: ' || member_name);
            DBMS_OUTPUT.PUT_LINE('Member_phone: ' || member_phone);
            DBMS_OUTPUT.PUT_LINE('Member_Add_Date: ' || TO_CHAR(member_date, 'YYYY-MM-
DD'));
            DBMS_OUTPUT.PUT_LINE('Member_Role: ' || member_role);
            DBMS_OUTPUT.PUT_LINE(' ');
            EXIT;
        ELSE
            M_exsist := FALSE;
        END IF;
    END LOOP;
    close c_member;

    IF M_exsist = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Member ID NOT Exists Try New');
    END IF;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--view the table inserted values

EXECUTE VIEWSINGLEMEMBERS('M-002');
EXECUTE VIEWSINGLEMEMBERS('CAPJ');
```

DELETE PROCEDURE

```
--Delete single Member (Procedure)
Create OR REPLACE PROCEDURE DeleteMember(M_Id VARCHAR)
IS
   cursor c_member IS SELECT
Member_Id,MEMBER_NAME,MEMBER_PHONE,MEMBER_ADD_DATE,MEMBER_ROLE from Member;
   member_id VARCHAR(100);
   member_name VARCHAR(100);
   member_phone VARCHAR(10);
   member_date DATE;
   member_role VARCHAR(100);
   M_exsist BOOLEAN := FALSE;
BEGIN
   OPEN c_member;
   LOOP
      FETCH c_member into member_id,member_name,member_phone,member_date,member_role;
      EXIT WHEN c_member%NOTFOUND;
      IF member_id = M_Id THEN
         M_exsist := TRUE;
         DBMS_OUTPUT.PUT_LINE(' ');
         DBMS_OUTPUT.PUT_LINE('Member Information:');
         DBMS_OUTPUT.PUT_LINE('Member_Id: ' || member_id);
         DBMS_OUTPUT.PUT_LINE('Member_Name: ' || member_name);
         DBMS_OUTPUT.PUT_LINE('Member_phone: ' || member_phone);
         DBMS_OUTPUT.PUT_LINE('Member_Add_Date: ' || TO_CHAR(member_date, 'YYYY-MM-DD'));
         DBMS_OUTPUT.PUT_LINE('Member_Role: ' || member_role);
         Delete from Member where Member_Id = M_Id;
         DBMS_OUTPUT.PUT_LINE('Member Deleted');
         EXIT;
      ELSE
         M_exsist := FALSE;
```

```
    END IF;
  END LOOP;
  close c_member;

  IF M_exsist = FALSE THEN
    DBMS_OUTPUT.PUT_LINE('Member ID NOT Exists Try New');
  END IF;

  EXCEPTION
  WHEN no_data_found THEN
    DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--view the table inserted values
EXECUTE DELETEMEMBER('M-002');
```

3.BORROWED

INSERT PL/SQL PROCEDURE CODE

```
-- Insert a Borrowed (Procedure)
CREATE OR REPLACE PROCEDURE insertBorrowed(Borrow_Id VARCHAR,M_ID VARCHAR,B_Id
VARCHAR,B_add_date DATE,B_Return_date DATE)
IS
    cursor c_borrow IS SELECT BORROWED_ID FROM BORROWED;
    existing_borrow_id VARCHAR(100);
    b_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO existing_borrow_id;
        EXIT WHEN c_borrow%NOTFOUND;
```

```
        IF existing_borrow_id = Borrow_Id THEN
            b_exsist := TRUE;
            EXIT;
        END IF;
    END LOOP;
    CLOSE c_borrow;

    IF b_exsist THEN
        DBMS_OUTPUT.PUT_LINE('Borrowed ID Already Exists. Try New.');
    ELSE
        INSERT INTO BORROWED(BORROWED_ID, MEMBER_ID, BOOK_ID, Borrowed_date,
RETURN_DATE, BOOK_RETURNED)
        VALUES(Borrow_Id, M_ID, B_Id, B_add_date, B_Return_date, 'N');
        DBMS_OUTPUT.PUT_LINE('Borrowed Inserted Complete');
        UPDATE BOOK SET BOOK_COPIES = BOOK.BOOK_COPIES - 1  where Book.BOOK_ID = B_ID ;
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-Insert the values using execute keyword hardcode values
EXECUTE INSERTBORROWED('B-002','M-001','B-002',DATE'2024-10-23',DATE'2024-10-24');
EXECUTE INSERTBORROWED('B-001','M-001','B-002',DATE'2024-10-15',DATE'2024-10-20');
```

Method = 2 using Prompt

```
--M2-Insert values using prompt
SET SERVEROUTPUT ON
ACCEPT Borrowed_id char PROMPT 'Enter Borrowed ID: Start B-'
ACCEPT Borrowed_Member_ID char PROMPT 'Enter Borrowed Member ID: M-'
ACCEPT Borrowed_Book_ID char PROMPT 'Enter Borrowed Book ID: B-'
ACCEPT Borrowed_date DATE PROMPT 'Enter Borrowed date: YYYY/MM/DD'
ACCEPT Return_date DATE PROMPT 'Enter Return Date: YYYY-MM-DD'
DECLARE
    Borrowedid Borrowed.BORROWED_ID%TYPE;
    BorrowedMemberID Borrowed.MEMBER_ID%TYPE;
```

```
    BorrowedBookID Borrowed.BOOK_ID%TYPE;
    Borroweddate Borrowed.BORROWED_DATE%TYPE;
    Returndate Borrowed.RETURN_DATE%TYPE;
BEGIN
    Borrowedid :='&Borrowed_id';
    BorrowedMemberID :='&Borrowed_Member_ID';
    BorrowedBookID :='&Borrowed_Book_ID';
    Borroweddate := TO_DATE('&Borrowed_date','YYYY-MM-DD');
    Returndate :='&Return_date';
    insertBorrowed(Borrowedid,BorrowedMemberID,BorrowedBookID,Borroweddate,Returndate);
END;
```

UPDATE PROCEDURE

```
-- Update a Borrowed (Procedure)
CREATE OR REPLACE PROCEDURE updateBorrowed(Borrow_Id VARCHAR,M_ID VARCHAR,B_Id
VARCHAR,B_add_date DATE,B_Return_date DATE,r_Returned char)
IS
    cursor c_borrow IS SELECT BORROWED_ID FROM BORROWED;
    existing_borrow_id VARCHAR(100);
    b_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO existing_borrow_id;
        EXIT WHEN c_borrow%NOTFOUND;
        IF existing_borrow_id = Borrow_Id THEN
            b_exsist := TRUE;
            EXIT;
        END IF;
    END LOOP;
    CLOSE c_borrow;

    IF b_exsist THEN
        UPDATE BORROWED SET MEMBER_ID=M_ID,BOOK_ID=B_Id, Borrowed_date=B_add_date,
RETURN_DATE=B_Return_date, Book_Returned = r_Returned  where BORROWED_ID=Borrow_Id;
        if r_Returned = 'Y' THEN
        UPDATE BOOK SET BOOK_COPIES = BOOK.BOOK_COPIES  + 1  where Book.BOOK_ID = B_ID
;
        end if;
        DBMS_OUTPUT.PUT_LINE('Borrowed Update Complete');
    ELSE

        DBMS_OUTPUT.PUT_LINE('Borrowed ID not Exists. Try New.');
    END IF;
```

```
EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
--M1-update the values using execute keyword hardcode values
EXECUTE updateBorrowed('B-001','M-001','B-001',DATE'2024-10-25',DATE'2024-10-24','N');
EXECUTE updateBorrowed('B-001','M-001','B-002',DATE'2024-10-23',DATE'2024-10-24','Y');
```

Method = 2 using Prompt

```
--M2-update values using prompt
SET SERVEROUTPUT ON
ACCEPT Borrowed_id char PROMPT 'Enter Borrowed ID: Start B-'
ACCEPT Borrowed_Member_ID char PROMPT 'Enter Borrowed Member ID: M-'
ACCEPT Borrowed_Book_ID char PROMPT 'Enter Borrowed Book ID: B-'
ACCEPT Borrowed_date DATE PROMPT 'Enter Borrowed date: YYYY/MM/DD'
ACCEPT Return_date DATE PROMPT 'Enter Return Date: YYYY-MM-DD'
ACCEPT Book_returned char PROMPT 'Enter Borrowed Book Returned Y/N:'
DECLARE
    Borrowedid Borrowed.BORROWED_ID%TYPE;
    BorrowedMemberID Borrowed.MEMBER_ID%TYPE;
    BorrowedBookID Borrowed.BOOK_ID%TYPE;
    Borroweddate Borrowed.BORROWED_DATE%TYPE;
    Returndate Borrowed.RETURN_DATE%TYPE;
    Bookreturned Borrowed.Book_returned%TYPE;
BEGIN
    Borrowedid :='&Borrowed_id';
    BorrowedMemberID :='&Borrowed_Member_ID';
    BorrowedBookID :='&Borrowed_Book_ID';
    Borroweddate := TO_DATE('&Borrowed_date','YYYY-MM-DD');
    Returndate :='&Return_date';
    Bookreturned :='&Book_returned';
    updateBorrowed(Borrowedid,BorrowedMemberID,BorrowedBookID,Borroweddate,Returndate,B
ookreturned);
```

```
END;
```

VIEW PROCEDURE

VIEW ALL

```
--View all details
CREATE OR REPLACE PROCEDURE ViewAllBorrowed
IS
    cursor c_borrow IS SELECT
BORROWED_ID,MEMBER_ID,BOOK_ID,BORROWED_DATE,RETURN_DATE,BOOK_RETURNED FROM BORROWED;
    existing_borrow_id VARCHAR(100);
    member_id VARCHAR(100);
    book_id VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    book_return char;
BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO
existing_borrow_id,member_id,book_id,borrowed_date,return_date,book_return;
        EXIT WHEN c_borrow%NOTFOUND;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
            DBMS_OUTPUT.PUT_LINE('Borrow_Id: ' || existing_borrow_id);
            DBMS_OUTPUT.PUT_LINE('Member_ID: ' || member_id);
            DBMS_OUTPUT.PUT_LINE('Book_id: ' || book_id);
            DBMS_OUTPUT.PUT_LINE('Borrowed_date: ' || borrowed_date);
            DBMS_OUTPUT.PUT_LINE('Return_date: ' || return_date);
            DBMS_OUTPUT.PUT_LINE('Book_return: ' || book_return);
            DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_borrow;
EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE VIEWALLBORROWED;
```

VIEW USING ID SPECIFIC

```
-- View single Borrowed (Procedure)
CREATE OR REPLACE PROCEDURE ViewSingleBorrowed(Borrow_Id VARCHAR)
IS
    cursor c_borrow IS SELECT BORROWED_ID,MEMBER_ID,BOOK_ID,BORROWED_DATE,RETURN_DATE
FROM BORROWED;
    existing_borrow_id VARCHAR(100);
    member_id VARCHAR(100);
    book_id VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    b_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO
existing_borrow_id,member_id,book_id,borrowed_date,return_date;
        EXIT WHEN c_borrow%NOTFOUND;
        IF existing_borrow_id = Borrow_Id OR member_id=Borrow_Id THEN
            b_exsist := TRUE;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
            DBMS_OUTPUT.PUT_LINE('Borrow_Id: ' || existing_borrow_id);
            DBMS_OUTPUT.PUT_LINE('Member_ID: ' || member_id);
            DBMS_OUTPUT.PUT_LINE('Book_id: ' || book_id);
            DBMS_OUTPUT.PUT_LINE('Borrowed_date: ' || borrowed_date);
            DBMS_OUTPUT.PUT_LINE('Return_date: ' || return_date);
            DBMS_OUTPUT.PUT_LINE(' ');
            EXIT;
        END IF;
    END LOOP;
    CLOSE c_borrow;

    IF b_exsist=FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Borrowed ID NOT Exists. Try New.');

    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE VIEWSINGLEBORROWED('B-002');
```

DELETE PROCEDURE

```
-- Delete single Borrowed (Procedure)
CREATE OR REPLACE PROCEDURE DeleteBorrowed(Borrow_Id VARCHAR)
IS
    cursor c_borrow IS SELECT BORROWED_ID,MEMBER_ID,BOOK_ID,BORROWED_DATE,RETURN_DATE
FROM BORROWED;
    existing_borrow_id VARCHAR(100);
    member_id VARCHAR(100);
    book_id VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    b_exsist BOOLEAN := FALSE;
    book_return char;
BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO
existing_borrow_id,member_id,book_id,borrowed_date,return_date, book_return;
        EXIT WHEN c_borrow%NOTFOUND;
        IF existing_borrow_id = Borrow_Id THEN
            b_exsist := TRUE;
            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
            DBMS_OUTPUT.PUT_LINE('Borrow_Id: ' || existing_borrow_id);
            DBMS_OUTPUT.PUT_LINE('Member_ID: ' || member_id);
            DBMS_OUTPUT.PUT_LINE('Book_id: ' || book_id);
            DBMS_OUTPUT.PUT_LINE('Borrowed_date: ' || borrowed_date);
            DBMS_OUTPUT.PUT_LINE('Return_date: ' || return_date);
            DBMS_OUTPUT.PUT_LINE('Book_return: ' || book_return);
            Delete from BORROWED where BORROWED_ID = Borrow_Id;
            DBMS_OUTPUT.PUT_LINE('Borrowed details Deleted');
            EXIT;
        END IF;
    END LOOP;
    CLOSE c_borrow;
```

```
    IF b_exsist=FALSE THEN
        DBMS_OUTPUT.PUT_LINE('Borrowed ID NOT Exists. Try New.');

    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE DELETEBORROWED('B-001');
```

USER ROLE

DISPLAY USER ROLES

```
--Select user roles and display
Create OR REPLACE PROCEDURE MembersRoles
IS
   cursor c_member IS SELECT Member_Id,MEMBER_ROLE from Member;
   member_id VARCHAR(100);
   member_role VARCHAR(100);
   M_exsist BOOLEAN := FALSE;
BEGIN
   OPEN c_member;
   LOOP
      FETCH c_member into member_id,member_role;
      EXIT WHEN c_member%NOTFOUND;

         DBMS_OUTPUT.PUT_LINE(' ');
         DBMS_OUTPUT.PUT_LINE('Member ROLE Information:');
         DBMS_OUTPUT.PUT_LINE('Member_Id: ' || member_id);
```

```
        DBMS_OUTPUT.PUT_LINE('Member_Role: ' || member_role);
        DBMS_OUTPUT.PUT_LINE(' ');
        EXIT;

    END LOOP;
    close c_member;

    EXCEPTION
    WHEN no_data_found THEN
      DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE MembersRoles;
```

REPORTS GENERATED AND PL/SQL CODE

1.AVAILABLE ALL BOOKS DETAILS

```
--1.Available Books Report

CREATE OR REPLACE PROCEDURE BookReport
IS
    CURSOR c_book IS
        SELECT Book_Id, BOOK_TITLE, BOOK_AUTHOR, BOOK_ADD_DATE, BOOK_COPIES
        FROM BOOK
        ORDER BY BOOK_ADD_DATE DESC;

    book_id VARCHAR(100);
    book_title VARCHAR(100);
    book_author VARCHAR(100);
    book_add_date DATE;
    book_copies NUMBER;

BEGIN
```

```
    OPEN c_book;
    LOOP
        FETCH c_book INTO book_id, book_title, book_author, book_add_date, book_copies;
        EXIT WHEN c_book%NOTFOUND;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Book Information:');
        DBMS_OUTPUT.PUT_LINE('Book_Id: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book_Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book_Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Book_Add_Date: ' || TO_CHAR(book_add_date, 'YYYY-MM-
DD'));
        DBMS_OUTPUT.PUT_LINE('Book_Copies: ' || book_copies);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_book;

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE BOOKREPORT;
```

2.LAST WEEK BORROWED BOOKS DETAILS

```
--2.last Week Borrowed Books Report

CREATE OR REPLACE PROCEDURE BorrowedReport
IS
    cursor c_borrow IS
        SELECT BORROWED.BOOK_ID, BORROWED.MEMBER_ID, BORROWED.BORROWED_DATE,
BORROWED.RETURN_DATE,
```

```
                Member.Member_name AS member_name, Member.Member_phone AS member_phone,
                Book.Book_Title AS book_title, Book.Book_Author AS book_author
        FROM BORROWED
        JOIN Member ON Borrowed.MEMBER_ID = Member.Member_Id
        JOIN Book ON Borrowed.Book_ID = Book.BOOK_ID
        Where Borrowed.Borrowed_date >= SYSDATE - 7 ;

    book_id VARCHAR(100);
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(100);
    book_title VARCHAR(100);
    book_author VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    Borrowed_available BOOLEAN := FALSE;

BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO book_id, member_id, borrowed_date, return_date,
member_name, member_phone, book_title, book_author;
        EXIT WHEN c_borrow%NOTFOUND;

        Borrowed_available := TRUE;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Member ID: ' || member_id);
        DBMS_OUTPUT.PUT_LINE('Member Name: ' || member_name);
        DBMS_OUTPUT.PUT_LINE('Member Phone: ' || member_phone);
        DBMS_OUTPUT.PUT_LINE('Borrowed Date: ' || borrowed_date);
        DBMS_OUTPUT.PUT_LINE('Return Date: ' || return_date);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_borrow;

    IF Borrowed_available = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('There are NO Borrowed Books To Return');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ' || SQLERRM);
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE BorrowedReport;
```

3.OVERDUE BOOKS DETAILS

```
CREATE OR REPLACE PROCEDURE OverDueReport
IS
    cursor c_borrow IS
        SELECT BORROWED.BOOK_ID, BORROWED.MEMBER_ID, BORROWED.BORROWED_DATE,
BORROWED.RETURN_DATE,
               Member.Member_name AS member_name, Member.Member_phone AS member_phone,
               Book.Book_Title AS book_title, Book.Book_Author AS book_author
        FROM BORROWED
        JOIN Member ON Borrowed.MEMBER_ID = Member.Member_Id
        JOIN Book ON Borrowed.Book_ID = Book.BOOK_ID
        WHERE Borrowed.Return_Date < SYSDATE and borrowed.Book_returned = 'N';

    book_id VARCHAR(100);
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(100);
    book_title VARCHAR(100);
    book_author VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    Borrowed_available BOOLEAN := FALSE;

BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO book_id, member_id, borrowed_date, return_date,
member_name, member_phone, book_title, book_author;
        EXIT WHEN c_borrow%NOTFOUND;

        Borrowed_available := TRUE;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
```

```
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Member ID: ' || member_id);
        DBMS_OUTPUT.PUT_LINE('Member Name: ' || member_name);
        DBMS_OUTPUT.PUT_LINE('Member Phone: ' || member_phone);
        DBMS_OUTPUT.PUT_LINE('Borrowed Date: ' || borrowed_date);
        DBMS_OUTPUT.PUT_LINE('Return Date: ' || return_date);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_borrow;

    IF Borrowed_available = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('There are NO OverDue Books To Return');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ' || SQLERRM);
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE OverDueReport;
```

4.MEMBERS DETAILS

```
--4.Members Report

Create OR REPLACE PROCEDURE MembersReport
IS
    cursor c_member IS SELECT
Member_Id,MEMBER_NAME,MEMBER_PHONE,MEMBER_ADD_DATE,MEMBER_ROLE from Member
    ORDER BY Member.MEMBER_ADD_DATE DESC;
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(10);
    member_date DATE;
    member_role VARCHAR(100);
    M_exsist BOOLEAN := FALSE;
BEGIN
    OPEN c_member;
```

```
    LOOP
        FETCH c_member into member_id,member_name,member_phone,member_date,member_role;
        EXIT WHEN c_member%NOTFOUND;

            DBMS_OUTPUT.PUT_LINE(' ');
            DBMS_OUTPUT.PUT_LINE('Member Information:');
            DBMS_OUTPUT.PUT_LINE('Member_Id: ' || member_id);
            DBMS_OUTPUT.PUT_LINE('Member_Name: ' || member_name);
            DBMS_OUTPUT.PUT_LINE('Member_phone: ' || member_phone);
            DBMS_OUTPUT.PUT_LINE('Member_Add_Date: ' || TO_CHAR(member_date, 'YYYY-MM-
DD'));
            DBMS_OUTPUT.PUT_LINE('Member_Role: ' || member_role);
            DBMS_OUTPUT.PUT_LINE(' ');
            EXIT;

    END LOOP;
    close c_member;

    EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrived from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURED');
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE MembersReport;
```

5.DAILY BORROWED BOOKS DETAILS

```
--5.Daily Borrowed Books report



CREATE OR REPLACE PROCEDURE BorrowedDailyReport
IS
    cursor c_borrow IS
        SELECT BORROWED.BOOK_ID, BORROWED.MEMBER_ID, BORROWED.BORROWED_DATE,
BORROWED.RETURN_DATE,
               Member.Member_name AS member_name, Member.Member_phone AS member_phone,
               Book.Book_Title AS book_title, Book.Book_Author AS book_author
```

```
        FROM BORROWED
        JOIN Member ON Borrowed.MEMBER_ID = Member.Member_Id
        JOIN Book ON Borrowed.Book_ID = Book.BOOK_ID
        WHERE TRUNC(Borrowed.Borrowed_date) = TRUNC(SYSDATE)
        ORDER BY BORROWED_DATE DESC;

    book_id VARCHAR(100);
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(100);
    book_title VARCHAR(100);
    book_author VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    Borrowed_available BOOLEAN := FALSE;

BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO book_id, member_id, borrowed_date, return_date,
member_name, member_phone, book_title, book_author;
        EXIT WHEN c_borrow%NOTFOUND;

        Borrowed_available := TRUE;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Member ID: ' || member_id);
        DBMS_OUTPUT.PUT_LINE('Member Name: ' || member_name);
        DBMS_OUTPUT.PUT_LINE('Member Phone: ' || member_phone);
        DBMS_OUTPUT.PUT_LINE('Borrowed Date: ' || borrowed_date);
        DBMS_OUTPUT.PUT_LINE('Return Date: ' || return_date);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_borrow;

    IF Borrowed_available = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('There are NO Borrowed Books Today');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ' || SQLERRM);
```

```
END;
```

EXECUTE THE PROCEDURE

Method = 1 Use Execute Key Word

```
EXECUTE BorrowedDailyReport;
```

6.MEMBER BORROWED BOOK HISTORY REPORT

```
--6.member borrowed history reoprt


CREATE OR REPLACE PROCEDURE member_borrow_history
IS
    cursor c_borrow IS
        SELECT BORROWED.BOOK_ID, BORROWED.MEMBER_ID, BORROWED.BORROWED_DATE,
BORROWED.RETURN_DATE,
               Member.Member_name AS member_name, Member.Member_phone AS member_phone,
               Book.Book_Title AS book_title, Book.Book_Author AS book_author
        FROM BORROWED
        JOIN Member ON Borrowed.MEMBER_ID = Member.Member_Id
        JOIN Book ON Borrowed.Book_ID = Book.BOOK_ID
        ORDER BY Member_Id DESC;

    book_id VARCHAR(100);
    member_id VARCHAR(100);
    member_name VARCHAR(100);
    member_phone VARCHAR(100);
    book_title VARCHAR(100);
    book_author VARCHAR(100);
    borrowed_date DATE;
    return_date DATE;
    Borrowed_available BOOLEAN := FALSE;

BEGIN
    OPEN c_borrow;
    LOOP
        FETCH c_borrow INTO book_id, member_id, borrowed_date, return_date,
member_name, member_phone, book_title, book_author;
        EXIT WHEN c_borrow%NOTFOUND;
```

```
        Borrowed_available := TRUE;

        DBMS_OUTPUT.PUT_LINE(' ');
        DBMS_OUTPUT.PUT_LINE('Borrowed Information:');
        DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_id);
        DBMS_OUTPUT.PUT_LINE('Book Title: ' || book_title);
        DBMS_OUTPUT.PUT_LINE('Book Author: ' || book_author);
        DBMS_OUTPUT.PUT_LINE('Member ID: ' || member_id);
        DBMS_OUTPUT.PUT_LINE('Member Name: ' || member_name);
        DBMS_OUTPUT.PUT_LINE('Member Phone: ' || member_phone);
        DBMS_OUTPUT.PUT_LINE('Borrowed Date: ' || borrowed_date);
        DBMS_OUTPUT.PUT_LINE('Return Date: ' || return_date);
        DBMS_OUTPUT.PUT_LINE(' ');
    END LOOP;
    CLOSE c_borrow;

    IF Borrowed_available = FALSE THEN
        DBMS_OUTPUT.PUT_LINE('There are NO Borrowed Books Today');
    END IF;

EXCEPTION
    WHEN no_data_found THEN
        DBMS_OUTPUT.PUT_LINE('NO data retrieved from the query');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('AN EXCEPTION OCCURRED: ' || SQLERRM);
END;
```

EXECUTE METHOD

```
EXECUTE member_borrow_history;
```

DATABASE ADMINISTRATION

CREATE USER

```
--Create admin and user roles

CREATE USER C##admin IDENTIFIED BY 12345;

CREATE USER C##user IDENTIFIED BY 12345;
```

GRANT PERMISSONS

```
--Grand permissions for the roles

GRANT ALL ON Book TO C##admin;
GRANT ALL ON Borrowed TO C##admin;
GRANT ALL ON Member TO C##admin;


GRANT SELECT ON Book TO C##user;
GRANT SELECT ON Borrowed TO C##user;
GRANT SELECT ON Member TO C##user;
```

BACKUP PLANS

- Cold backups : Collect the backup when users are not using the LMS.(database shutdown period).
- Hot Backup : Online Backup while database used by user. Database must in the ARCHIVELOG mode.
- RMAN : Tool for incremental backup and corruption detection.

Full Backup for Monthly Backup Plan and Incremental Backup for Weekly to reduce the data loss.

Full Backup Plan : Helps to get all the data store safe and recover easily.

Incremental Backup Plan : Use to store updated data store for a while until full backup get.

FULL BACKUP CODE USING VS-CODE

```
--Full Backup VS CODE

Create OR Replace DIRECTORY backup_folder As 'location';
GRANT ALL ON DIRECTORY backup_folder TO system;
```

```
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

FULL BACKUP USING CMD

```
--FULL BACKUP USING CMD
/*
rman target /
Backup DATABASE format 'location.bkp';
SHUTDOWN IMMEDIATE;
STARTUP MOUNT;
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
BacKUP DATABSE FORMAT 'location.bkp';
*/
```

CLOUD PLATFORM

- Automate the maintains for high performance and scalability.
- Maintain a single database to run across multiple servers for availability.
- Google cloud provide a multi-cloud experience with Vertex Ai support.
- Cloud provides a simplifies operations and deployment.

DATA SECURITY

- Secure Login access.
- Encrypt the sensitive data.
- Mask the Sensitive data.
- Restrict the permissions for users accessing sensitive data.
- Continuous monitoring.

REFERENCES

cellularnews. (n.d.). Retrieved from cellularnews: https://cellularnews.com/definitions/what-is-oracle-database-oracle-db/

datamation. (n.d.). Retrieved from datamation: https://www.datamation.com/big-data/oracle-database-rdbms/

educba. (n.d.). Retrieved from educba: https://www.educba.com/what-is-oracle/

scaler. (n.d.). Retrieved from scaler: https://www.scaler.com/topics/postgresql-vs-mysql/

sprinkledata. (n.d.). Retrieved from sprinkledata: https://www.sprinkledata.com/blogs/mongodb-vs-oracle-a-comparative-analysis-of-two-leading-database-systems

PROJECT GIT LINK

https://github.com/Dhanushanandan/LMS_DBMS-2.git