**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**

[**DECLARATION** 3](#_Toc181387133)

[**INTRODUCTION** 6](#_Toc181387134)

[INTRODUCTIONS TO BUSINESS 6](#_Toc181387135)

[**REASON TO USE ORACLE DATABASE** 6](#_Toc181387136)

[**NEED FOR AN ORACLE DATBASE FOR LMS** 7](#_Toc181387137)

[**REQUIREMENTS OF ANALYSIS** 7](#_Toc181387138)

[FUNCTIONAL REQUIREMENTS 7](#_Toc181387139)

[NON-FUNCTIONAL REQUIREMENTS 8](#_Toc181387140)

[**DATABASE DESIGN** 8](#_Toc181387141)

[LOGICAL DESIGN 8](#_Toc181387142)

[PHYSICAL DESIGN 8](#_Toc181387143)

[DATABASE TABLE CREATE CODE 9](#_Toc181387144)

[**CRUD OPERATIONS FOR THE LMS** 10](#_Toc181387145)

[**1.BOOK** 10](#_Toc181387146)

[INSERT PL/SQL PROCEDURE CODE 10](#_Toc181387147)

[UPDATE PROCEDURE 12](#_Toc181387148)

[VIEW PROCEDURE 14](#_Toc181387149)

[DELETE PROCEDURE 16](#_Toc181387150)

[**2.MEMBER** 17](#_Toc181387151)

[INSERT PL/SQL PROCEDURE CODE 17](#_Toc181387152)

[UPDATE PROCEDURE 19](#_Toc181387153)

[VIEW PROCEDURE 21](#_Toc181387154)

[DELETE PROCEDURE 24](#_Toc181387155)

[**3.BORROWED** 25](#_Toc181387156)

[INSERT PL/SQL PROCEDURE CODE 25](#_Toc181387157)

[UPDATE PROCEDURE 27](#_Toc181387158)

[VIEW PROCEDURE 29](#_Toc181387159)

[DELETE PROCEDURE 31](#_Toc181387160)

[**USER ROLE** 32](#_Toc181387161)

[DISPLAY USER ROLES 32](#_Toc181387162)

[**REPORTS GENERATED AND PL/SQL CODE** 33](#_Toc181387163)

[1.AVAILABLE ALL BOOKS DETAILS 33](#_Toc181387164)

[2.LAST WEEK BORROWED BOOKS DETAILS 34](#_Toc181387165)

[3.OVERDUE BOOKS DETAILS 36](#_Toc181387166)

[4.MEMBERS DETAILS 37](#_Toc181387167)

[5.DAILY BORROWED BOOKS DETAILS 38](#_Toc181387168)

[**DATABASE ADMINISTRATION** 41](#_Toc181387169)

[CREATE USER 41](#_Toc181387170)

[GRANT PERMISSONS 42](#_Toc181387171)

[**BACKUP PLANS** 42](#_Toc181387172)

[FULL BACKUP CODE USING VS-CODE 42](#_Toc181387173)

[FULL BACKUP USING CMD 43](#_Toc181387174)

[**CLOUD PLATFORM** 43](#_Toc181387175)

[**DATA SECURITY** 43](#_Toc181387176)

[**REFERENCES** 44](#_Toc181387177)

[**PROJECT GIT LINK** 44](#_Toc181387178)

# **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\_ROLE=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,Bookreturned);

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