LIBRARY MANAGEMENT SYSTEM

VANITA VISHRAM WOMEN'S UNIVERSITY

Managed by Vanita Vishram, Surat

PREPARED BY: -

PATIL POOJA NARAYAN(2312189)

SUPERVISOR: -

DR. HEMANGINI PATEL

VANITA VISHRAM WOMEN'S UNIVERSITY
Managed By Vanita Vishram, Surat

School of Science and Technology

Department Of Computer Science

Program: BCA

2024 - 2025

CERTIFICATE

This is to certify that Miss_____of Class_____, Roll No_____has satisfactorily completed her Practical \ Laboratory Work for SYBCA Semester 3 in subject of Relational Database Management System for the term ending in Month October and year 2024 - 2025.

Date:

Subject Expert 1	**Subject Expert 2**
Name:	**Name:**
_____	_____
Signature:	**Signature:**
_____	_____

VANITA VISHRAM WOMEN'S UNIVERSITY
Managed By Vanita Vishram, Surat

School of Science and Technology
Department of Computer Science
Program: BCA

Laboratory \ Practical Index

Sr. No.	Particulars	Practical Date	Page No	Signature

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of this Library Management System project.

First and foremost, we extend our sincere thanks to our project supervisor, [Supervisor's Name], for their invaluable guidance and support throughout the project. Their insights and expertise have been instrumental in shaping our understanding and approach.

We would also like to acknowledge the efforts of our team members, whose collaboration and dedication made this project possible. Special thanks to [Team Member Names] for their hard work and commitment in developing and refining the system.

Additionally, we appreciate the support from the library staff and users who provided essential feedback and requirements that greatly enhanced the system's design and functionality.

Finally, we would like to thank our families and friends for their encouragement and understanding during the project. This endeavor would not have been possible without the collective support and efforts of everyone involved. Thank you!

INTRODUCTION TO DATABASE PROJECT

This project focuses on designing and implementing a database system that involves the development of two key phases. The primary goal of this project is to ensure seamless data management, processing, and retrieval by structuring the database efficiently and implementing robust PL/SQL components.

Phase I: Table Structure Design

In the first phase, the emphasis is on creating an optimal table structure, which forms the foundation of the database. Key tasks in this phase include defining the relationships between tables and determining how data will be stored and accessed. The design focuses on the normalization of the tables to avoid redundancy and ensure data integrity.

Phase II: PL/SQL Implementation

In the second phase, the project advances into developing PL/SQL components such as stored procedures, functions, and triggers. These elements are essential for automating database operations, ensuring data consistency, and improving the performance of the system. The PL/SQL scripts enhance the interaction between the user and the database by allowing for efficient query execution and error handling.

Project Objectives

- **Efficient Data Management:** Design a database with optimized relationships between tables to manage data effectively.
- **Data Integrity & Consistency:** Implement PL/SQL components to ensure that data is accurate, reliable, and adheres to business rules.
- **Scalability:** Develop a scalable database structure that can handle an increasing volume of data and queries.
- **Automation & Performance:** Use stored procedures, functions, and triggers to automate repetitive tasks, enhance database performance, and ensure smooth operations.

This database project serves as a comprehensive solution for managing data efficiently while ensuring integrity and performance through well-designed table structures and PL/SQL components.

ADVANTAGES

The Library Management System (LMS) project you described offers several advantages that can enhance library operations and user experience. Here are some key benefits:

1. Streamlined Operations

- Automation of Transactions: The system automates book borrowing, returning, and transaction management, reducing manual efforts and errors.
- Centralized Data Management: All data related to authors, publishers, books, members, and transactions is stored in a single database, making it easier to manage and retrieve information.

2. Improved User Experience

- User-Friendly Interface: A well-designed front-end allows members to search for books, check availability, and manage their accounts efficiently.
- Quick Access to Information: Members can quickly access their membership details, borrowed books, and transaction history, enhancing transparency.

3. Enhanced Data Integrity

- Referential Integrity: The use of foreign keys ensures that all references between tables are valid, maintaining data consistency across the system.
- Error Handling: PL/SQL blocks, triggers, and exception handling mechanisms prevent invalid data entries and ensure robust data management.

4. Comprehensive Reporting and Analytics

- Transaction Records: The system can generate reports on book transactions, member activity, and inventory, helping library staff make informed decisions.
- Statistical Analysis: Libraries can analyze book popularity, member engagement, and other metrics to improve services and resource allocation.

5. Flexible and Scalable Design

- Expandable Framework: The system can be expanded to include additional features such as online reservations, digital libraries, or integration with other library systems.
- Scalability: As the library grows, the database can accommodate more books, members, and transactions without a significant redesign.

6. Enhanced Security

- Access Control: Sensitive data can be protected through access controls, ensuring that only authorized personnel can perform certain actions, like modifying member records or managing transactions.
- Data Backup and Recovery: Implementing backup procedures ensures that data can be restored in case of failures or data loss.

7. Increased Efficiency in Resource Management

- Inventory Management: The system allows libraries to track book availability and manage their collection effectively.
- Member Management: It simplifies managing member registrations, renewals, and communications, enhancing member relationships.

8. Time-Saving Features

- Automated Notifications: The system can send automated notifications to members for due dates, renewals, and library events, saving staff time and improving member engagement.
- Quick Searching and Filtering: Advanced search features help users find books and information quickly, reducing wait times and enhancing user satisfaction.

9. Community Engagement

- Member Participation: By allowing members to track their borrowing history and favorite books, the system fosters greater engagement and participation in library activities.
- Feedback Mechanisms: The system can incorporate features for members to provide feedback, enabling libraries to continuously improve their services.

10. Cost-Effectiveness

- Reduction in Operational Costs: Automation of processes reduces the need for manual labor, leading to cost savings in the long term.
- Better Resource Allocation: With improved tracking and analytics, libraries can allocate resources more effectively, minimizing waste.

Requirement Determination and Analysis for a Library Management System

In this phase, we identify what the Library Management System (LMS) needs to do. First, we talk to library staff, users, and management to understand their needs. We gather requirements through interviews, surveys, and observations of current processes.

We categorize these requirements into functional (like user registration and book search) and non-functional (like usability and security). Then, we prioritize them to focus on the most important ones first. Finally, we validate the requirements by sharing them with stakeholders for feedback and create a clear document to guide the development of the system. This ensures that the LMS will effectively serve the library's needs.

Phasel

1. Create a Table with appropriate constraints (Primary Key, Foreign Key, etc...)

1.AUTHORS TABLE.

```
CREATE TABLE AUTHORS
(
    AUTHOR_ID INTEGER NOT NULL PRIMARY KEY,
    NAME VARCHAR2(200),
    DOJ DATE
);
```

DESC AUTHORS;

Object Type **TABLE** Object **AUTHORS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
AUTHORS	AUTHOR_ID	Number	-	-	0	1	-	-	-
	NAME	Varchar2	200	-	-	-	✓	-	-
	DOJ	Date	7	-	-	-	✓	-	-

1 - 3

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	AUTHOR_ID	NUMBER(38,0)	No	(null)	1	(null)
2	NAME	VARCHAR2(200 BYTE)	Yes	(null)	2	(null)
3	DOJ	DATE	Yes	(null)	3	(null)

2.PUBLISHERS TABLE.

```
CREATE TABLE PUBLISHERS
(
    PUBLISHER_ID INTEGER NOT NULL PRIMARY KEY,
    NAME VARCHAR2(200),
    ADDRESS VARCHAR2(200),
    AUTHOR_ID INTEGER,
    CONSTRAINT FK_A_P
    FOREIGN KEY (AUTHOR_ID)
    REFERENCES AUTHORS(AUTHOR_ID)
);
```

DESC PUBLISHERS;

Object Type **TABLE** Object **PUBLISHERS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PUBLISHERS	PUBLISHER_ID	Number	-	-	0	1	-	-	-
	NAME	Varchar2	200	-	-	-	✓	-	-
	ADDRESS	Varchar2	200	-	-	-	✓	-	-
	AUTHOR_ID	Number	-	-	0	-	✓	-	-

1 - 4

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PUBLISHER_ID	NUMBER(38,0)	No	(null)	1	(null)
2	NAME	VARCHAR2(200 BYTE)	Yes	(null)	2	(null)
3	ADDRESS	VARCHAR2(200 BYTE)	Yes	(null)	3	(null)
4	AUTHOR_ID	NUMBER(38,0)	Yes	(null)	4	(null)

3.BOOKS TABLE.

```
CREATE TABLE BOOKS
(
    BOOK_ID INTEGER NOT NULL PRIMARY KEY,
    AUTHOR_ID INTEGER NOT NULL,
    PUBLISHER_ID INTEGER NOT NULL,
    GENRE VARCHAR2(200),
    PUBLICATION_YEAR INTEGER,
    AVAILABLE_COPIES INTEGER,
    CONSTRAINT FK_A_B
    FOREIGN KEY (AUTHOR_ID)
    REFERENCES AUTHORS,
    CONSTRAINT FK_P_B
    FOREIGN KEY (PUBLISHER_ID)
    REFERENCES PUBLISHERS
);
```

DESC BOOKS;

Object Type **TABLE** Object **BOOKS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BOOKS	BOOK_ID	Number	-	-	0	1	-	-	-
	AUTHOR_ID	Number	-	-	0	-	-	-	-
	PUBLISHER_ID	Number	-	-	0	-	-	-	-
	GENRE	Varchar2	200	-	-	-	✓	-	-
	PUBLICATION_YEAR	Number	-	-	0	-	✓	-	-
	AVAILABLE_COPIES	Number	-	-	0	-	✓	-	-

1 - 6

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	BOOK_ID	NUMBER(38,0)	No	(null)	1	(null)
2	AUTHOR_ID	NUMBER(38,0)	No	(null)	2	(null)
3	PUBLISHER_ID	NUMBER(38,0)	No	(null)	3	(null)
4	GENRE	VARCHAR2(200 BYTE)	Yes	(null)	4	(null)
5	PUBLICATION_YEAR	NUMBER(38,0)	Yes	(null)	5	(null)
6	AVAILABLE_COPIES	NUMBER(38,0)	Yes	(null)	6	(null)

4.MEMBERS TABLE

```
CREATE TABLE MEMBERS
(
    MEMBER_ID INTEGER NOT NULL PRIMARY KEY,
     NAME VARCHAR2(200),
     MEMBERSHIP_DATE DATE,
     EMAIL VARCHAR2(200),
     BOOK_ID INTEGER NOT NULL,
     AUTHOR_ID INTEGER NOT NULL,
     PUBLISHER_ID INTEGER NOT NULL,
    CONSTRAINT FK_B_M
    FOREIGN KEY (BOOK_ID)
    REFERENCES BOOKS (BOOK_ID),
    CONSTRAINT Fk_PU_BO
    FOREIGN KEY (PUBLISHER_ID)
    REFERENCES PUBLISHERS (PUBLISHER_ID),
    CONSTRAINT FK_AU_BO
    FOREIGN KEY (AUTHOR_ID)
    REFERENCES AUTHORS(AUTHOR_ID)
 );
```

DESC MEMBERS;

Object Type **TABLE** Object **MEMBERS**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MEMBERS	MEMBER_ID	Number	-	-	0	1	-	-	-
	NAME	Varchar2	200	-	-	-	✓	-	-
	MEMBERSHIP_DATE	Date	7	-	-	-	✓	-	-
	EMAIL	Varchar2	200	-	-	-	✓	-	-
	BOOK_ID	Number	-	-	0	-	-	-	-
	AUTHOR_ID	Number	-	-	0	-	-	-	-
	PUBLISHER_ID	Number	-	-	0	-	-	-	-

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	MEMBER_ID	NUMBER(38,0)	No	(null)	1	(null)
2	NAME	VARCHAR2(200 BYTE)	Yes	(null)	2	(null)
3	MEMBERSHIP_DATE	DATE	Yes	(null)	3	(null)
4	EMAIL	VARCHAR2(200 BYTE)	Yes	(null)	4	(null)
5	BOOK_ID	NUMBER(38,0)	No	(null)	5	(null)
6	AUTHOR_ID	NUMBER(38,0)	No	(null)	6	(null)
7	PUBLISHER_ID	NUMBER(38,0)	No	(null)	7	(null)

5.TRANSACTIONS TABLE.

```
CREATE TABLE TRANSACTIONS
(
  TRANSACTION_ID INTEGER NOT NULL PRIMARY KEY,
  BOOK_ID INTEGER NOT NULL,
   MEMBER_ID INTEGER NOT NULL,
  PUBLISHER_ID INTEGER NOT NULL,
  AUTHOR_ID INTEGER NOT NULL,
  ISSUE_DATE DATE,
  RETURN_DATE DATE,
  DUE_DATE DATE,
  CONSTRAINT FK_B_T FOREIGN KEY (BOOK_ID) REFERENCES BOOKS(BOOK_ID),
  CONSTRAINT FK_M_T FOREIGN KEY (MEMBER_ID) REFERENCES MEMBERS(MEMBER_ID),
  CONSTRAINT FK_A_T FOREIGN KEY (AUTHOR_ID) REFERENCES AUTHORS(AUTHOR_ID),
   CONSTRAINT FK_P_T FOREIGNKEY(PUBLISHER_ID) REFERENCES PUBLISHERS(PUBLISHER_ID)
);
```

DESC TRANSACTIONS;

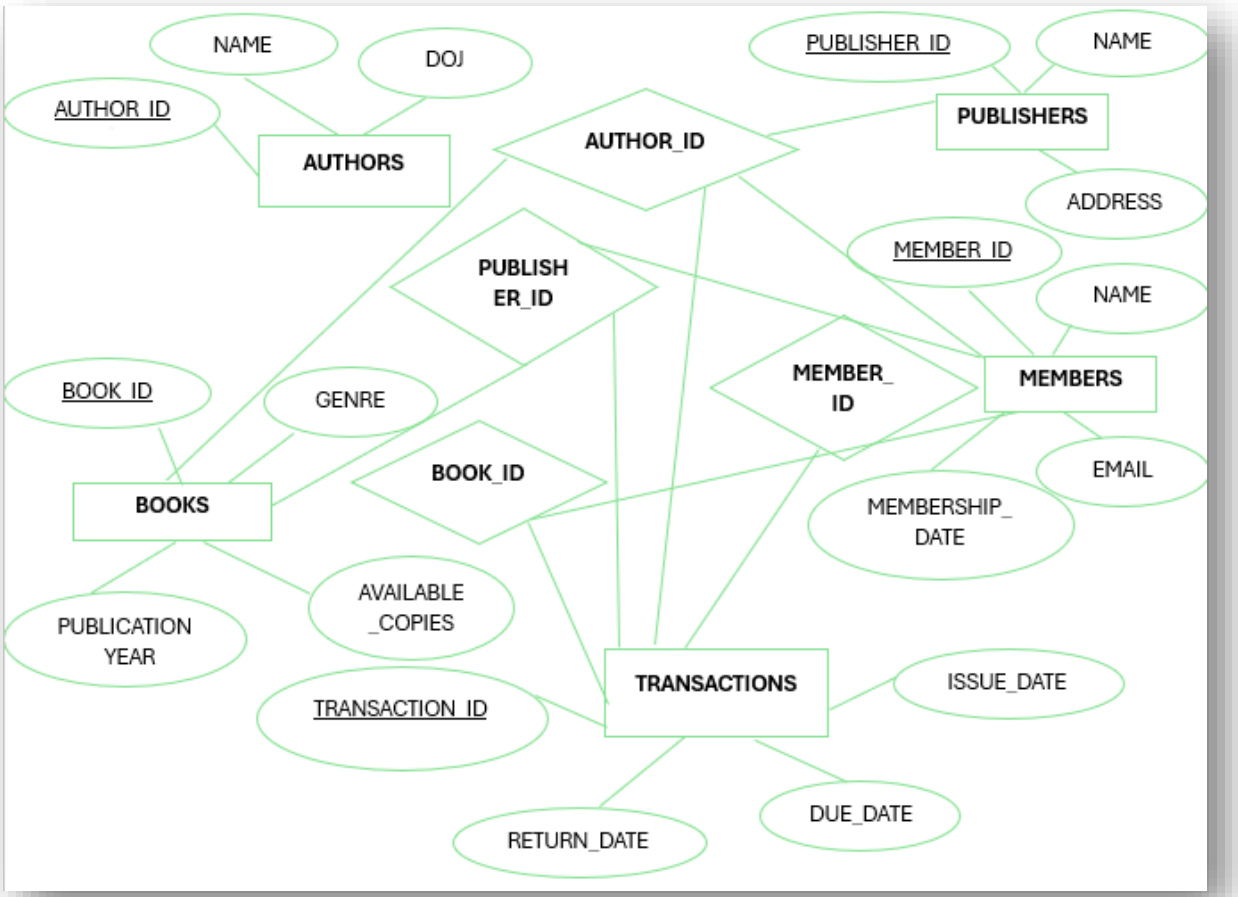
Object Type **TABLE** Object **TRANSACTIONS**

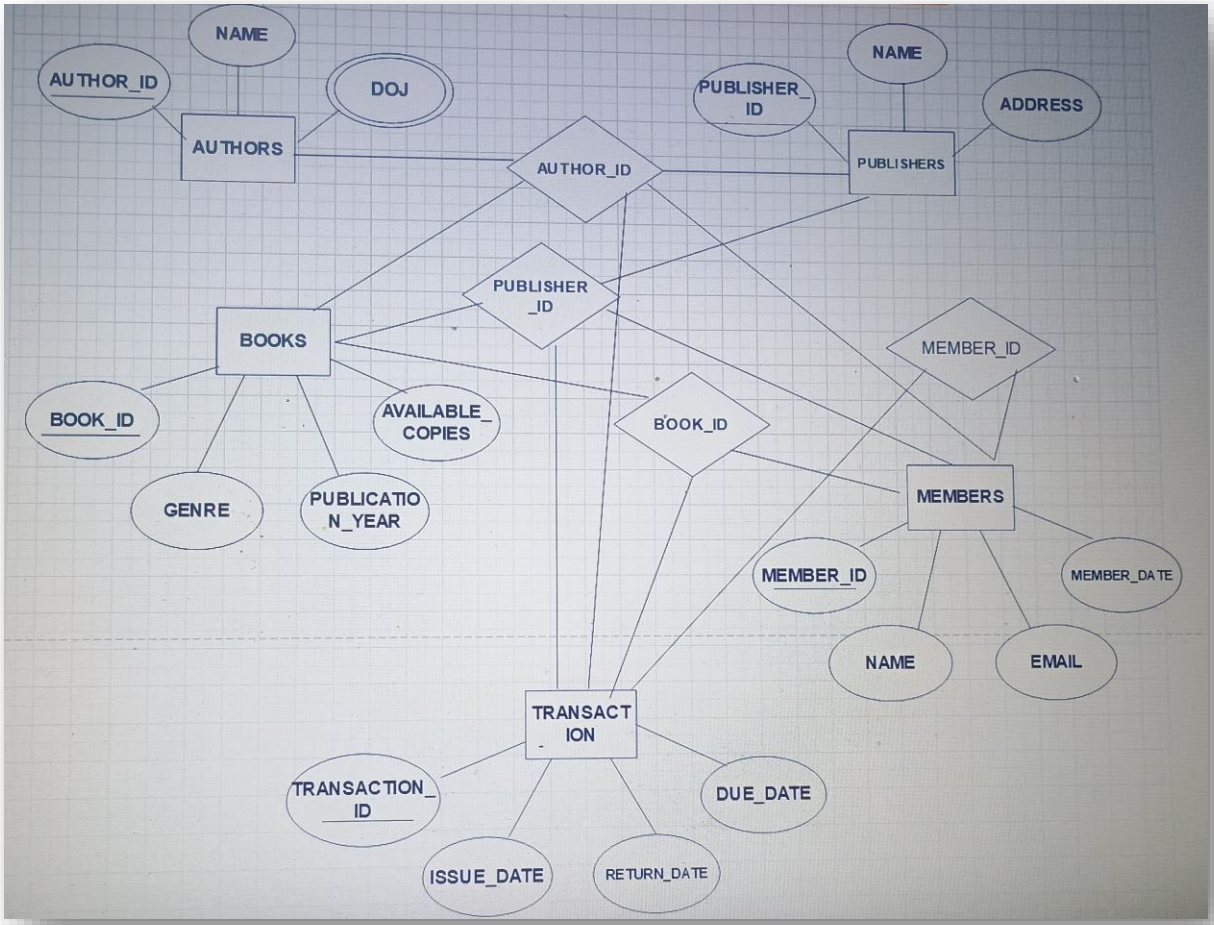
Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
TRANSACTIONS	TRANSACTION_ID	Number	-	-	0	1	-	-	-
	BOOK_ID	Number	-	-	0	-	-	-	-
	MEMBER_ID	Number	-	-	0	-	-	-	-
	PUBLISHER_ID	Number	-	-	0	-	-	-	-
	AUTHOR_ID	Number	-	-	0	-	-	-	-
	ISSUE_DATE	Date	7	-	-	-	✓	-	-
	RETURN_DATE	Date	7	-	-	-	✓	-	-
	DUE_DATE	Date	7	-	-	-	✓	-	-

1 - 8

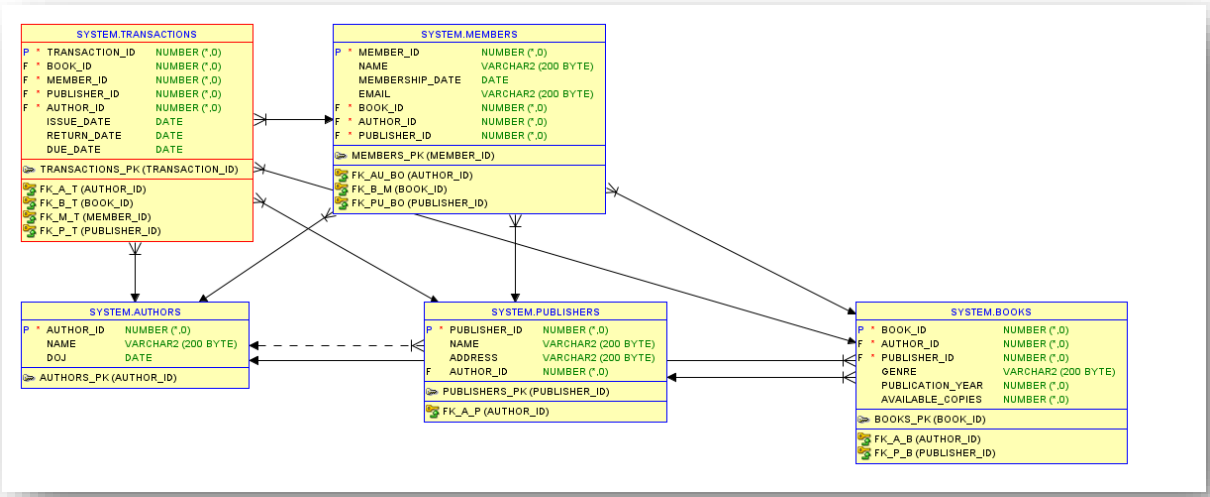
	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TRANSACTION_ID	NUMBER(38,0)	No	(null)	1	(null)
2	BOOK_ID	NUMBER(38,0)	No	(null)	2	(null)
3	MEMBER_ID	NUMBER(38,0)	No	(null)	3	(null)
4	PUBLISHER_ID	NUMBER(38,0)	No	(null)	4	(null)
5	AUTHOR_ID	NUMBER(38,0)	No	(null)	5	(null)
6	ISSUE_DATE	DATE	Yes	(null)	6	(null)
7	RETURN_DATE	DATE	Yes	(null)	7	(null)
8	DUE_DATE	DATE	Yes	(null)	8	(null)

2. Draw an E-R diagram for the same.




NAME
AUTHOR_ID
DOJ
AUTHORS
AUTHOR_ID
NAME
PUBLISHER_ID
ADDRESS
PUBLISHERS
PUBLISHER_ID
BOOKS
MEMBER_ID
BOOK_ID
AVAILABLE_COPIES
BOOK_ID
GENRE
PUBLICATION_YEAR
MEMBERS
MEMBER_ID
MEMBER_DATE
NAME
EMAIL
TRANSACTION
TRANSACTION_ID
DUE_DATE
ISSUE_DATE
RETURN_DATE

3. Table Relationship Diagram for same. (Use ORACLE SQL Developer)



Phase II (Using PL/SQL)

1. Use PL/SQL script to Insert appropriate record

SEQUENCE :

```
CREATE SEQUENCE SEQ_MEMBER
INCREMENT BY 1
START WITH 1
MINVALUE 1
MAXVALUE 100
NOCYCLE;
```

1.PL/SQL BLOCK FOR AUTHORS

```
DECLARE
AUTHOR_ID1 AUTHORS.AUTHOR_ID%TYPE;
NAME1 AUTHORS.NAME%TYPE;
DOJ1 AUTHORS.DOJ%TYPE;
BEGIN
AUTHOR_ID1:=:AUTHOR_ID1;
NAME1:=:NAME1;
DOJ1:=:DOJ1;
INSERT INTO AUTHORS VALUES (AUTHOR_ID1,NAME1,DOJ1);
END;
SELECT * FROM AUTHORS;
```

AUTHOR_ID	NAME	DOJ
1	George Orwell	25-JUN-03
2	J.K.Rowling	31-JUL-65
3	F.Scott Fitzgerald	24-SEP-96
4	Agatha Christie	15-SEP-90
5	J.R.R Tolkien	03-JAN-92

	AUTHOR_ID	NAME	DOJ
1	1	George Orwell	25-06-03
2	2	J.K.Rowling	31-07-65
3	3	F.Scott Fitzgerald	24-09-96
4	4	Agatha Christie	15-09-90
5	5	J.R.R Tolkien	03-01-92

2.PL/SQL BLOCK FOR PUBLISHERS

```
DECLARE
PUBLISHER_ID1 PUBLISHERS.PUBLISHER_ID%TYPE;
NAME1  PUBLISHERS.NAME%TYPE;
ADDRESS1 PUBLISHERS.ADDRESS%TYPE;
AUTHOR_ID1  PUBLISHERS.AUTHOR_ID%TYPE;
BEGIN
PUBLISHER_ID1:=:PUBLISHER_ID1;
NAME1:=:NAME1;
ADDRESS1:=:ADDRESS1;
AUTHOR_ID1:=:AUTHOR_ID1;
INSERT INTO PUBLISHERS VALUES(PUBLISHER_ID1,NAME1,ADDRESS1,AUTHOR_ID1);
END;
SELECT * FROM PUBLISHERS;
```

PUBLISHER_ID	NAME	ADDRESS	AUTHOR_ID
1	Penguin Books	Surat	1
2	Bloomsburry	Navsari	2
3	Scribner	Ahmedabad	3
4	HarperCollins	Gandhinagar	4
5	Allen & Unwin	Bardoli	5

	PUBLISHER_ID	NAME	ADDRESS	AUTHOR_ID
1	1	Penguin Books	Surat	1
2	2	Bloomsburry	Navsari	2
3	3	Scribner	Ahmedabad	3
4	4	HarperCollins	Gandhinagar	4
5	5	Allen & Unwin	Bardoli	5

3.PL/SQL BLOCK FOR BOOKS

```
DECLARE
BOOK_ID1 BOOKS.BOOK_ID%TYPE;
AUTHOR_ID1 BOOKS.AUTHOR_ID%TYPE;
PUBLISHER_ID1 BOOKS.PUBLISHER_ID%TYPE;
GENRE1 BOOKS.GENRE%TYPE;
PUBLICATION_YEAR BOOKS.PUBLICATION_YEAR%TYPE;
AVAILABLE_COPIES1 BOOKS.AVAILABLE_COPIES%TYPE;
BEGIN
BOOK_ID1:=:BOOK_ID1;
AUTHOR_ID1:=:AUTHOR_ID1;
PUBLISHER_ID1:=:PUBLISHER_ID1;
GENRE1:=:GENRE1;
PUBLICATION_YEAR:=:PUBLICATION_YEAR;
AVAILABLE_COPIES1:=:AVAILABLE_COPIES1;
INSERT INTO BOOKS
VALUES(BOOK_ID1,AUTHOR_ID1,PUBLISHER_ID1,GENRE1,PUBLICATION_YEAR,AVAILABLE_C
OPIES1);
END;
SELECT * FROM BOOKS;
```

BOOK_ID	AUTHOR_ID	PUBLISHER_ID	GENRE	PUBLICATION_YEAR	AVAILABLE_COPIES
1	1	1	Dystopian	1949	5
2	2	2	Fantasy	1997	10
3	3	3	Classic	1925	7
4	4	4	Mystery	1934	8
5	5	5	Fantasy	1954	15

	BOOK_ID	AUTHOR_ID	PUBLISHER_ID	GENRE	PUBLICATION_YEAR	AVAILABLE_COPIES
1	1	1	1	Dystopian	1949	5
2	2	2	2	Fantasy	1997	10
3	3	3	3	Classic	1925	7
4	4	4	4	Mystery	1934	8
5	5	5	5	Fantasy	1954	15

4.PL/SQL BLOCK FOR MEMBERS

```
DECLARE
  MEMBER_ID1 MEMBERS.MEMBER_ID%TYPE;
  NAME1 MEMBERS.NAME%TYPE;
  MEMBERSHIP_DATE1 MEMBERS.MEMBERSHIP_DATE%TYPE;
  EMAIL1 MEMBERS.EMAIL%TYPE;
  BOOK_ID1 MEMBERS.BOOK_ID%TYPE;
  AUTHOR_ID1 MEMBERS.AUTHOR_ID%TYPE;
  PUBLISHER_ID1 MEMBERS.PUBLISHER_ID%TYPE;
BEGIN
  MEMBER_ID1 := :MEMBER_ID1;
  NAME1 := :NAME1;
  MEMBERSHIP_DATE1 := :MEMBERSHIP_DATE1;
  EMAIL1 := :EMAIL1;
  BOOK_ID1 := :BOOK_ID1;
  AUTHOR_ID1 := :AUTHOR_ID1;
  PUBLISHER_ID1 := :PUBLISHER_ID1;
  INSERT INTO MEMBERS VALUES (MEMBER_ID1, NAME1, MEMBERSHIP_DATE1,
EMAIL1, BOOK_ID1, AUTHOR_ID1, PUBLISHER_ID1);
END;
SELECT * FROM MEMBERS;
```

MEMBER_ID	NAME	MEMBERSHIP_DATE	EMAIL	BOOK_ID	AUTHOR_ID	PUBLISHER_ID
1	ALICE SMITH	15-JAN-23	alice.smith@gmail.com	1	1	1
2	BOB JOHNSON	20-FEB-23	bob.johson@gmail.com	2	2	2
3	CAROL WHITE	03-OCT-23	carol.white@gmail.com	3	3	3
4	DAVID BROWN	05-MAY-23	david.brown@gmail.com	4	4	4
5	EVA GREEN	15-APR-23	eva.green@gmail.com	5	5	5

	MEMBER_ID	NAME	MEMBERSHIP_DATE	EMAIL	BOOK_ID	AUTHOR_ID	PUBLISHER_ID
1	1	ALICE SMITH	15-01-23	alice.smith@gmail.com	1	1	1
2	2	BOB JOHNSON	20-02-23	bob.johson@gmail.com	2	2	2
3	3	CAROL WHITE	03-10-23	carol.white@gmail.com	3	3	3
4	4	DAVID BROWN	05-05-23	david.brown@gmail.com	4	4	4
5	5	EVA GREEN	15-04-23	eva.green@gmail.com	5	5	5

5. PL/SQL BLOCK FOR TRANSACTIONS

```
DECLARE
TRANSACTION_ID1 TRANSACTIONS.TRANSACTION_ID%TYPE;
BOOK_ID1 TRANSACTIONS.BOOK_ID%TYPE;
MEMBER_ID1 TRANSACTIONS.MEMBER_ID%TYPE;
PUBLISHER_ID1 TRANSACTIONS.PUBLISHER_ID%TYPE;
AUTHOR_ID1 TRANSACTIONS.AUTHOR_ID%TYPE;
ISSUE_DATE1 TRANSACTIONS.ISSUE_DATE%TYPE;
RETURN_DATE1 TRANSACTIONS.RETURN_DATE%TYPE;
DUE_DATE1 TRANSACTIONS.DUE_DATE%TYPE;
BEGIN
TRANSACTION_ID1:=:TRANSACTION_ID1;
BOOK_ID1:=:BOOK_ID1;
MEMBER_ID1:=:MEMBER_ID1;
PUBLISHER_ID1:=:PUBLISHER_ID1;
AUTHOR_ID1:=:AUTHOR_ID1;
ISSUE_DATE1:=:ISSUE_DATE1;
RETURN_DATE1:=:RETURN_DATE1 ;
DUE_DATE1:=:DUE_DATE1;
INSERT INTO TRANSACTIONS
VALUES('T'||SEQ_MEMBER.NEXTVAL,BOOK_ID1,MEMBER_ID1,PUBLISHER_ID1,AUTHO
R_ID1,ISSUE_DATE1,RETURN_DATE1,DUE_DATE1);
END;
SELECT * FROM TRANSACTIONS;
```

TRANSACTION_ID	BOOK_ID	MEMBER_ID	PUBLISHER_ID	AUTHOR_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE
T1	1	1	1	1	01-JUL-24	15-JUL-24	16-JUL-24
T2	2	2	2	2	01-AUG-24	15-AUG-24	16-AUG-24
T3	3	3	3	3	01-SEP-24	15-SEP-24	16-SEP-24
T4	4	4	4	4	01-OCT-24	15-OCT-24	16-OCT-24
T5	5	5	5	5	01-NOV-24	15-NOV-24	16-NOV-24

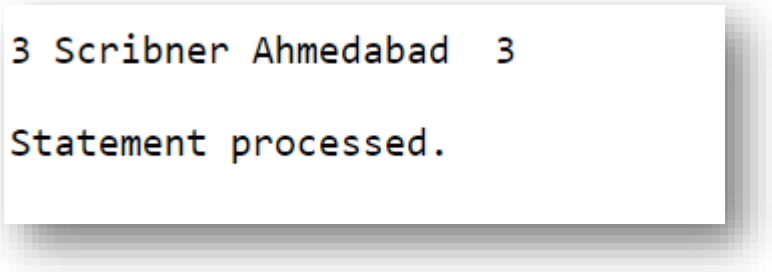
	TRANSACTION_ID	BOOK_ID	MEMBER_ID	PUBLISHER_ID	AUTHOR_ID	ISSUE_DATE	RETURN_DATE	DUE_DATE
1	1	1	1	1	1	01-07-24	15-07-24	16-07-24
2	2	2	2	2	2	01-08-24	15-08-24	16-08-24
3	3	3	3	3	3	01-09-24	15-09-24	16-09-24
4	4	4	4	4	4	01-10-24	15-10-24	16-10-24
5	5	5	5	5	5	01-11-24	15-11-24	16-11-24

2. Make one cursor program for project.

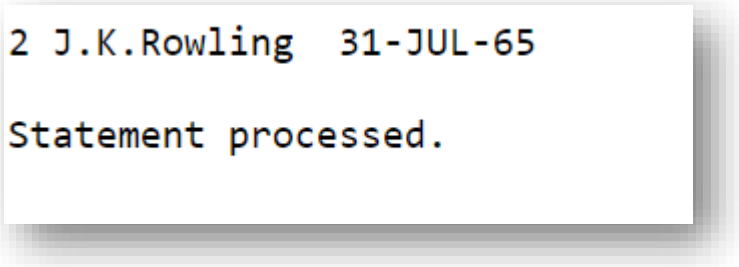
```
DECLARE

CURSOR C1 IS SELECT MEMBER_ID,
NAME,MEMBERSHIP_DATE,BOOK_ID,AUTHOR_ID,PUBLISHER_ID FROM MEMBERS;

X C1%ROWTYPE;

BEGIN

FOR X IN C1

LOOP

DBMS_OUTPUT.PUT_LINE(X.MEMBER_ID||' '|| X.NAME||' '||X.MEMBERSHIP_DATE||'
'||X.BOOK_ID||' '||X. AUTHOR_ID||' '|| X.PUBLISHER_ID);

END LOOP;

END;
```

```
1    ALICE SMITH    15-JAN-23     1  1  1
2    BOB JOHNSON    20-FEB-23     2  2  2
3    CAROL WHITE    03-OCT-23     3  3  3
4    DAVID BROWN    05-MAY-23     4  4  4
5    EVA GREEN    15-APR-23     5  5  5
```

```
DECLARE

CURSOR C1 IS SELECT PUBLISHER_ID ,NAME,ADDRESS, AUTHOR_ID FROM PUBLISHERS
WHERE AUTHOR_ID = 3;

X C1%ROWTYPE;

BEGIN

FOR X IN C1

LOOP

DBMS_OUTPUT.PUT_LINE(X.PUBLISHER_ID||' '|| X.NAME||' '||X.ADDRESS||'  '||X.AUTHOR_ID);

END LOOP;

END;
```

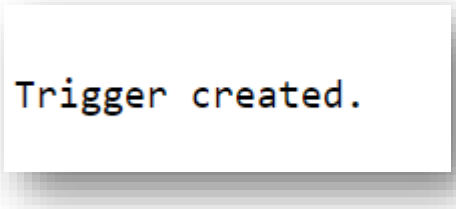


```
DECLARE
CURSOR C1 IS SELECT AUTHOR_ID, NAME, DOJ FROM AUTHORS WHERE DOJ = '31-JUL-1965';
X C1 %ROWTYPE;
BEGIN
FOR X IN C1
LOOP
DBMS_OUTPUT.PUT_LINE(X.AUTHOR_ID||' '|| X.NAME||' '||X.DOJ);
END LOOP;
END;
```



3. Make one trigger for the same.

```
CREATE OR REPLACE TRIGGER TR1
AFTER INSERT ON TRANSACTIONS
BEGIN
IF RTRIM(TO_CHAR(SYSDATE,'DAY'))='SUNDAY'
THEN
RAISE_APPLICATION_ERROR(-20001,'CANT INSERT THE RECORD ON SUNDAY');
END IF;
END;
```

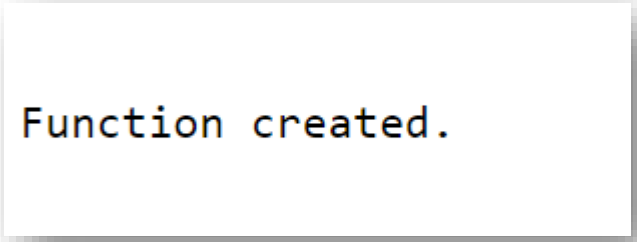


4. Make function and procedure for same.

FUNCTION

```
CREATE OR REPLACE FUNCTION SRCH_MEMBERS(MEMBER_ID IN
MEMBERS.MEMBER_ID%TYPE)
RETURN VARCHAR2
IS
 MEMBER_ID1 MEMBERS.MEMBER_ID%TYPE;
 NAME1 MEMBERS.NAME%TYPE;
 MEMBERSHIP_DATE1 MEMBERS.MEMBERSHIP_DATE%TYPE;
 EMAIL1 MEMBERS.EMAIL%TYPE;
 AUTHOR_ID1 MEMBERS.AUTHOR_ID%TYPE;
 RESULT VARCHAR2(4000); -- Variable to store result string
BEGIN
 SELECT MEMBER_ID, NAME, MEMBERSHIP_DATE, EMAIL, AUTHOR_ID
 INTO MEMBER_ID1, NAME1, MEMBERSHIP_DATE1, EMAIL1, AUTHOR_ID1
 FROM MEMBERS
 WHERE MEMBER_ID = SRCH_MEMBERS.MEMBER_ID; -- Compare with input parameter
 RESULT := MEMBER_ID1 || '  ' || NAME1 || '  ' ||
       TO_CHAR(MEMBERSHIP_DATE1, 'YYYY-MM-DD') || '  ' || EMAIL1 || '  ' || AUTHOR_ID1;
           DBMS_OUTPUT.PUT_LINE(RESULT);
          RETURN RESULT;

EXCEPTION
 WHEN NO_DATA_FOUND THEN
  DBMS_OUTPUT.PUT_LINE('NO ROW IS SELECTED');
  RETURN 'NO ROW IS SELECTED';
 WHEN TOO_MANY_ROWS THEN
  DBMS_OUTPUT.PUT_LINE('MULTIPLE ROWS CANNOT BE SELECTED');
  RETURN 'MULTIPLE ROWS CANNOT BE SELECTED';
 WHEN VALUE_ERROR THEN
  DBMS_OUTPUT.PUT_LINE('MEMORY INSUFFICIENT');
```

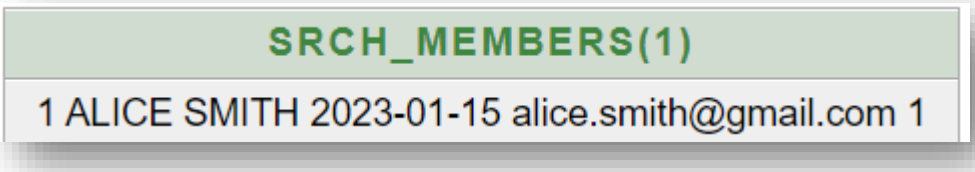
```
  RETURN 'MEMORY INSUFFICIENT';
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('AN ERROR OCCURRED: ' || SQLERRM);
  RETURN 'AN ERROR OCCURRED: ' || SQLERRM;
END;
```



EXECUTING A FUNCTION

1) **USING SELECT STATEMENT.**

```
SELECT SRCH_MEMBERS(1) FROM DUAL;
```



2) **USING MAIN PROGRAM.**

```
DECLARE
 RESULT VARCHAR2(4000);
BEGIN
 RESULT := SRCH_MEMBERS(1);
 RESULT := SRCH_MEMBERS(2);
 RESULT := SRCH_MEMBERS(3);
 RESULT := SRCH_MEMBERS(4);
 RESULT := SRCH_MEMBERS(5);
END;
```

```
1  ALICE SMITH  2023-01-15  alice.smith@gmail.com  1
2  BOB JOHNSON  2023-02-20  bob.johson@gmail.com  2
3  CAROL WHITE  2023-10-03  carol.white@gmail.com  3
4  DAVID BROWN  2023-05-05  david.brown@gmail.com  4
5  EVA GREEN  2023-04-15  eva.green@gmail.com  5

Statement processed.
```

PROCEDURE

```
CREATE OR REPLACE PROCEDURE SRCH_MEMBER(MEMBER_ID IN
MEMBERS.MEMBER_ID%TYPE)

I

 MEMBER_ID1 MEMBERS.MEMBER_ID%TYPE;

 NAME1 MEMBERS.NAME%TYPE;

 MEMBERSHIP_DATE1 MEMBERS.MEMBERSHIP_DATE%TYPE;

 EMAIL1 MEMBERS.EMAIL%TYPE;

 AUTHOR_ID1 MEMBERS.AUTHOR_ID%TYPE;

BEGIN


 SELECT MEMBER_ID, NAME, MEMBERSHIP_DATE, EMAIL, AUTHOR_ID

 INTO MEMBER_ID1, NAME1, MEMBERSHIP_DATE1, EMAIL1, AUTHOR_ID1

 FROM MEMBERS

 WHERE MEMBER_ID = SRCH_MEMBER.MEMBER_ID;


 DBMS_OUTPUT.PUT_LINE(MEMBER_ID1 || ' ' || NAME1 || ' ' || MEMBERSHIP_DATE1 || ' ' ||
EMAIL1 || ' ' || AUTHOR_ID1);


EXCEPTION

 WHEN NO_DATA_FOUND THEN

  DBMS_OUTPUT.PUT_LINE('NO ROW IS SELECTED');

 WHEN TOO_MANY_ROWS THEN

  DBMS_OUTPUT.PUT_LINE('MULTIPLE ROWS CAN''T BE SELECTED');

 WHEN VALUE_ERROR THEN

  DBMS_OUTPUT.PUT_LINE('MEM INSUFFICIENT');

END;
```

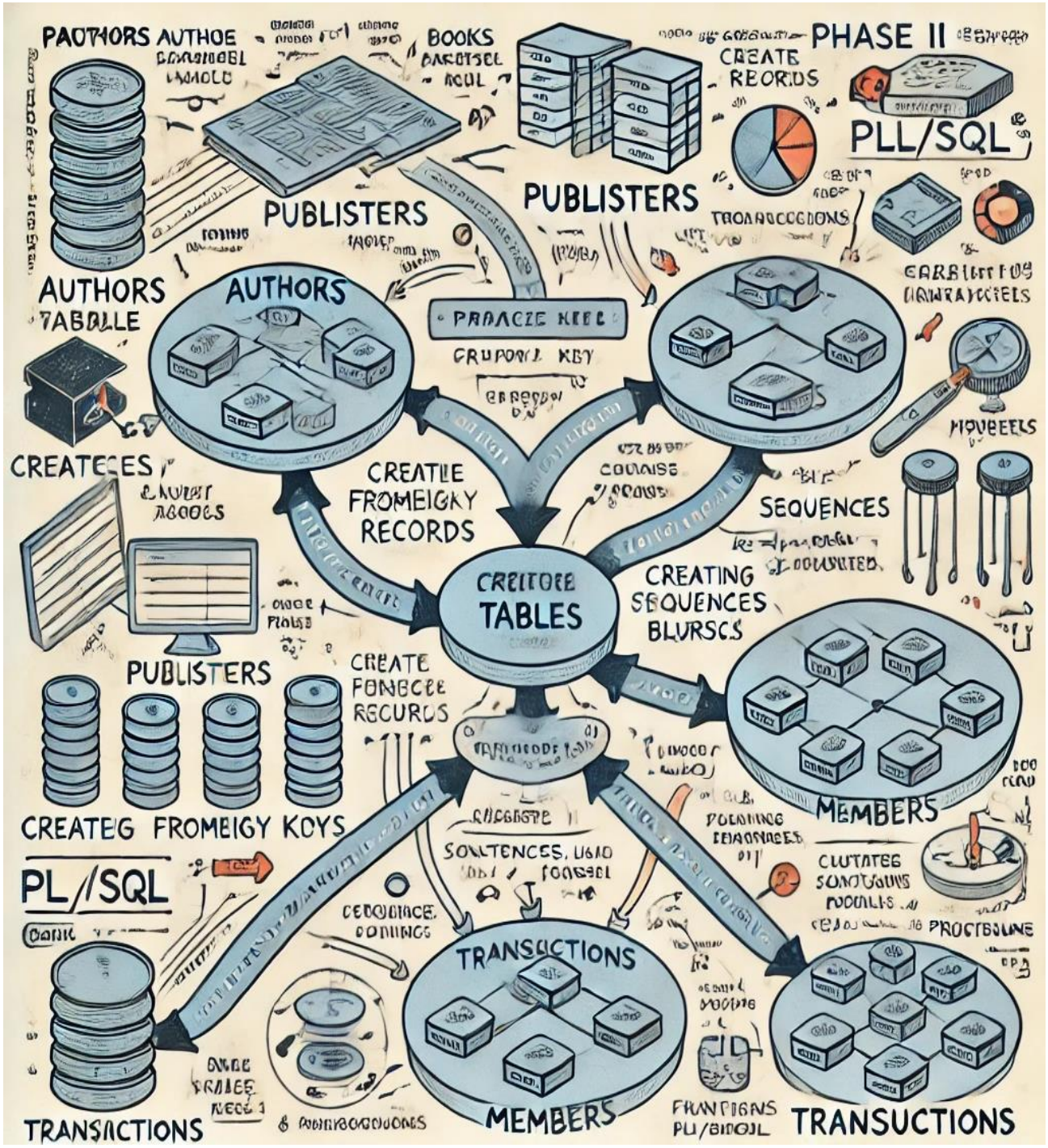
```
Procedure created.
```

EXECUTING A PROCEDURE

```
DECLARE

MEMBER_ID INTEGER :=:MEMBER_ID;

BEGIN

DBMS_OUTPUT.PUT_LINE('DETAILS OF MEMBER-ID '||MEMBER_ID);

DBMS_OUTPUT.PUT_LINE('MEMBER_ID'||' '||'NAME'||' '||'MEMBERSHIP_DATE'||' '||'EMAIL'||'
'||'AUTHOR_ID');

SRCH_MEMBER(MEMBER_ID);

END;
```

```
DETAILS OF MEMBER-ID 1
MEMBER_ID       NAME       MEMBERSHIP_DATE       EMAIL     AUTHOR_ID
1     ALICE SMITH     15-JAN-23     alice.smith@gmail.com     1

Statement processed.
```

MIND MAP OF LIBRBARY MANAGEMENT SYSTEM



Conclusion

The library management system designed in this project effectively addresses the core functionalities required to manage books, authors, publishers, members, and transactions within a library. By using Oracle SQL and PL/SQL, the database schema enforces strong relationships between tables through the use of primary keys, foreign keys, and constraints, ensuring data integrity and consistency.

The system allows seamless operations such as recording author details, book publications, member information, and transactions like book issues and returns. The E-R diagram and relationship diagrams provide clear visualizations of the data structure and relationships between entities such as authors, publishers, books, members, and transactions.

The implementation of PL/SQL blocks for data insertion ensures the efficient management of records with the help of sequences for generating unique IDs. Furthermore, triggers prevent invalid transactions, such as prohibiting record insertion on Sundays, which adds to the business rules and logic of the system.

The use of cursors, triggers, functions, and procedures provides powerful tools to handle various business requirements like fetching specific member information, automating transaction processes, and maintaining the overall functionality of the library system. With robust error handling, the system is capable of managing exceptional situations, making it more reliable.

In conclusion, this library management system offers a structured, efficient, and scalable solution for managing a library's operations, with future enhancements possible through the flexible use of PL/SQL for additional functionalities.