Library management system

ER diagram

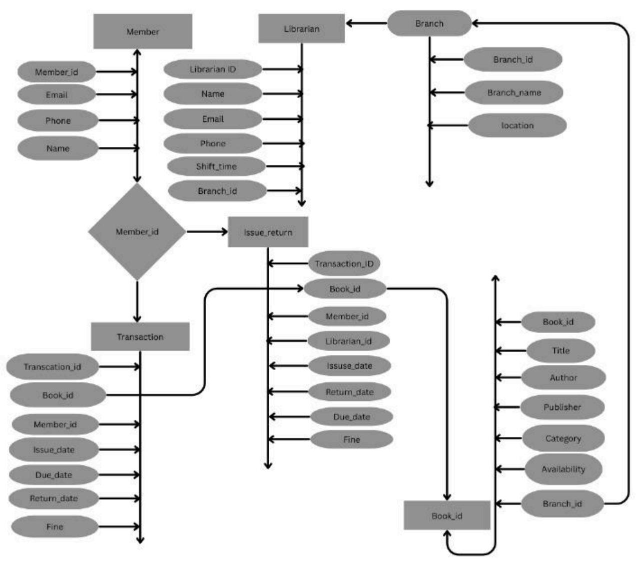

Table Creation and Insertion:

Table 1

SQL> create table members(member_id NUMBER(20) PRIMARY KEY,name VARCHAR2(20) NOT NULL,email VARCHAR2(30) UNIQUE NOT NULL, salary NUMBER(30) CHECK (salary > 0),phone NUMBER(30) UNIQUE NOT NULL);

Table created.

SQL>insert into members values('0001','nitharshana','nithu@gmail.com','60000','9876543210');

1 row created.

```
SQL> insert into members values('0002','navis
evangelin','navis@gmail.com','50000','8765432109');

1 row created.

SQL> insert into members
values('0003','nidharshini','nidharshini@gmail.com','40000','7654321098');

1 row created.
SQL> insert into members
values('0004','nirupa','nirupa@gmail.com','50000','6543210987');

1 row created.
```

Table 2

```
CREATE TABLE transaction_table ( transaction_id NUMBER(20) PRIMARY KEY, book_id
NUMBER(20) NOT NULL,  member_id      NUMBER(30) NOT NULL, staff_id       NUMBER(20),
sell_date      VARCHAR(20),  due_date       VARCHAR(30),  selling_price  NUMBER(30)
CHECK (selling_price > 0) );
Table created.

INSERT INTO transaction_table
VALUES (101, 1001, 10001, 100001, '2025-01-10', '2025-01-15', 150);

INSERT INTO transaction_table
VALUES (102, 1002, 10002, 100002, '2025-02-02', '2025-02-23', 200);
1 row created.
```

Table 3

```
CREATE TABLE issue_return (transaction_id INT PRIMARY KEY,book_id        INT NOT
NULL,    member_id      INT NOT NULL,   librarian_id   INT NOT NULL,  issue_date
DATE NOT NULL,return_date    DATE, due_date       DATE NOT NULL,fine_decimal
DECIMAL(10, 2) CHECK (fine_decimal >= 0));

Table created.
```

```
SQL> INSERT INTO issue_return   VALUES (101, 1001, 10001, 100001, DATE '2025-01-01',
DATE '2025-01-10', DATE '2025-01-15', 15);

1 row created.
 INSERT INTO issue_return VALUES (102,1002,10002,100002,DATE
'2025-02-02',DATE '2025-02-02',);

1 row created.
```

Table 4

```
CREATE TABLE branch ( branch_id    NUMBER(30) PRIMARY KEY,   branch_name
VARCHAR2(30) NOT NULL,   location     VARCHAR2(30) NOT NULL );

Table created.

SQL> insert into branch values('0001','centeal city library','coimbatore');

1 row created.

SQL> insert into branch values('0002','westside branch','chennai');

1 row created.

SQL> insert into branch values('0003','north campus library','erode');

1 row created.

SQL> insert into branch values('0004','south branch','salem');

1 row created.
```

Table 5

```
SQL> CREATE TABLE BOOKS (BOOK_ID VARCHAR2(20),TITLE VARCHAR2(20),AUTHOR
VARCHAR2(20),PUBLISHER VARCHAR2(30),CATEGORY VARCHAR2(40),AVAILABILITY
VARCHAR2(20), BRANCH_ID VARCHAR2(20));

Table created.

SQL> INSERT INTO BOOKS VALUES ('002', 'IKIGAI', 'SRIYOSA', 'AJ PUBLICATIONS',
'PSYCHOLOGY', '10', '21345');
```

```
1 row created.

SQL> INSERT INTO BOOKS VALUES ('003', 'ATOMIC HABITS', 'JAMES CLEAR', 'RR
PUBLICATIONS', 'MOTIVATIONAL', '20', '45560');

1 row created.

SQL> INSERT INTO BOOKS VALUES ('004', 'DEEP WORK', 'CAL NEWPORT', 'ER
PUBLICATIONS', 'PRODUCTIVITY', '12', '47860');
1 row created.
```

Table 6

```
SQL> CREATE TABLE LIBRARIANS (LIBRARIAN_ID VARCHAR2(10) PRIMARY KEY, NAME
VARCHAR2(50) NOT NULL, EMAIL   VARCHAR2(50) UNIQUE NOT NULL,PHONE
VARCHAR2(15) UNIQUE NOT NULL,SHIFT_TIME   VARCHAR2(20) CHECK (SHIFT_TIME IN
('Morning', 'Evening', 'Night')),BRANCH_ID    VARCHAR2(10) NOT NULL);

Table created

SQL> INSERT INTO LIBRARIANS
VALUES('L001','NIRUPA','nirupa@example.com','9876543210','Morning', 'B001'  );

1 row created.

SQL> INSERT INTO LIBRARIANS VALUES
('L002','NIDHARSHANA','nitharshana@example.com','9123456780','Evening', 'B002');


1 row created.

SQL>INSERT INTO LIBRARIANS   VALUES ('L003', 'NIDHARSHINI',
'nidharshini@example.com', '9012345678', 'Morning', 'B001');

1 row created.

SQL> INSERT INTO LIBRARIANS VALUES ('L004','NAVIS
EVANGELIN','navis.evangelin@example.com','9988776655','Night','B003');

1 row created.
```

DATA DEFINITION LANGUAGE

1)Add a new column Return_Status to the issue_return table to store return status text up to 20 characters.

```
SQL>ALTER TABLE issue_return ADD Return_Status VARCHAR(20);

TABLE ALTERED
```

2)Add a new column `remarks` to hold additional comments or notes, up to 100 characters.

```
SQL>ALTER TABLE issue_return ADD remarks VARCHAR(100);
TABLE ALTERED
```

3)Add a new column `late_days` to store the number of days a return is late as an integer.

```
SQL>ALTER TABLE issue_return ADD late_days INT;

TABLE ALTERED
```

4) Add a new column `date_of_birth` to the `Members` table to store each member's birth date in `DATE` format.

```
SQL>ALTER TABLE Members
ADD date_of_birth DATE;

TABLE ALTERED
```

5)Changes the phone column in the Members table to allow up to 20 characters, typically to support longer or international phone numbers.

```
SQL>ALTER TABLE Members
MODIFY phone VARCHAR(20);

TABLE ALTERED
```

6)Renames the column address in the Members table to residential_address for better clarity or specificity.

```
SQL>ALTER TABLE Members
RENAME COLUMN address TO residential_address;
```

TABLE ALTERED

7)Removes the `date_of_birth` column from the `Members` table, permanently deleting its data and structure.

```
SQL>ALTER TABLE Members
DROP COLUMN date_of_birth;
```

TABLE ALTERED

8) Permanently deletes the Members table and all of its data from the database.

```
SQL>DROP TABLE Members;
```

TABLE ALTERED

9)Deletes all data in the table but keeps the structure.

```
SQL>TRUNCATE TABLE Members;
```

TABLE ALTERED

10)Renames the table Members to LibraryMembers, typically to reflect a more specific or updated purpose.

```
SQL>RENAME TABLE Members TO LibraryMembers;
```

TABLE ALTERED

11)Permanently removes all rows from the "book_transactions" table, resetting it to empty but keeping its structure intact.

```
SQL>TRUNCATE TABLE book_transactions;
```

12) Permanently deletes the entire "book_transactions" table, including all its data and structure, from the database.

```
SQL>DROP TABLE book_transactions;
```

DATA MANIPULATION LANGUAGE

1)Retrieves all columns and all rows from the "Members" table.

```
SQL>SELECT * FROM Members;
```

2)Updates the email, phone, and address of the member with member_id equal to 1 in the "Members" table.

```
SQL>UPDATE Members
SET email = 'john.newemail@example.com', phone = '1112223333', address
= '789 Oak St, City'
     WHERE member_id = 1;
```

3)Deletes the member from the "Members" table whose member_id is 1

```
SQL>SELETE FROM Members WHERE member_id = 1;
```

4)Inserts a new row into the "Members" table with the values

```
SQL> insert into members
values('0004','nirupa','nirupa@gmail.com','50000','6543210987');
```

DATA INTEGRITY CONSTRAINTS

1)Adds a primary key constraint named PK_Salary on the Salary_ID column of the Salary table, ensuring each value in that column is unique and not null.

```
SQL>ALTER TABLE Salary
ADD CONSTRAINT PK_Salary PRIMARY KEY (Salary_ID);
```

2)Adds a primary key constraint named pk_branch on the branch_id column in the Branch table to ensure each branch_id is unique and not null.

```
SQL>ALTER TABLE Branch ADD CONSTRAINT pk_branch PRIMARY KEY(branch_id);

Table altered.
```

3)Adds a unique constraint named unique_branch_name on the branch_name column in the Branch table, ensuring all branch names are unique and no duplicates are allowed.

```
SQL> ALTER TABLE Branch ADD CONSTRAINT unique_branch_name
UNIQUE(branch_name); Table altered.
```

4)Adds a unique constraint named unique_branch_combination on the combination of branch_name and location columns in the Branch table, ensuring no two rows have the same pair of branch name and location.

```
SQL>ALTER TABLE Branch ADD CONSTRAINT unique_branch_combination
UNIQUE(branch_name,location);

Table altered.
```

5)Modifies the location column in the branch table to make it of type VARCHAR2(30) and ensures it cannot contain NULL values.

```
SQL>ALTER TABLE branch MODIFY location VARCHAR2(30)NOT NULL;

 Table altered.
```

6)Adds a check constraint named chk_location to the branch table, allowing only the values 'coimbatore', 'chennai', 'erode', or 'salem' in the location column.

```
SQL>ALTER TABLE branch ADD CONSTRAINT chk_location CHECK(location IN
('coimbatore','chennai','erode','salem'));

Table altered.
```

TRANSACTION CONTROL LANGUAGE

1)Commit the transaction and immediately start a new one:

```
SQL>COMMIT AND CHAIN;

TRANSACTION COMMITED
```

2)Commit the current transaction, making all changes permanent, and immediately start a new one:

```
    SQL>COMMIT AND BEGIN;
```

3)End the current transaction with a savepoint and continue with the next transaction:

```
    SQL>RELEASE savepoint_stefan AND CHAIN;
```

4)Commit the current transaction to make all changes permanent and end the transaction:

```
SQL>COMMIT;

COMMIT COMPLETE
```

5)Commit a specific savepoint within a transaction and continue with the transaction:

```
SQL>COMMIT TO savepoint_shubman;

SAVEPOINT COMMITED
```

6)Commit the transaction and immediately start a new one:

```
SQL>COMMIT AND CHAIN;

TRANSACTION COMMITED
```

7)Rollback to the start of the transaction, undoing all changes, and end the transaction:

```
SQL>ROLLBACK TO START;
```

8)Set a savepoint within a transaction for a specific point:

```
SQL>SAVEPOINT savepoint_tiger;

SAVEPOINT TIGER ESTABLISHED
```

9)Set a savepoint within a transaction and specify a name:

```
SQL>SAVEPOINT custom_savepoint;
```

10)Release a specific savepoint within a transaction:

```
SQL>RELEASE savepoint_virat;

SAVEPOINT RELEASED
```

11)Release a custom savepoint within a transaction:

```
SQL>RELEASE custom_savepoint;
```

12)Start a new transaction explicitly:

```
SQL>BEGIN;
```

13)Start a new transaction with a custom name:

```
SQL>BEGIN WORK;
```

14)Commit the current transaction, making all changes permanent, and immediately start a new one:

```
SQL>COMMIT AND BEGIN;
```

DATA QUERY LANGUAGE

1)Command to retrieve all member names from the members table:

```
SQL> SELECT name FROM members;

NAME
-------------------------
nitharshana
navis evangelin
nidharshini
nirupa
```

2)Retrieve all member details:

```
SQL> SELECT * FROM members;
```

3)Retrieve email addresses of all members:

```
SQL> SELECT email FROM members;
```

4) Retrieve members with a salary greater than 45,000:

```
SQL> SELECT * FROM members WHERE salary > 45000;
```

5)Retrieve names and phone numbers of members with salary = 50000:

```
SQL> SELECT name, phone FROM members WHERE salary = 50000;
```

6) Retrieve the total number of members:

```
SQL> SELECT COUNT(*) FROM members;
```

7) Retrieve names of members whose name starts with 'n':

```
SQL> SELECT name FROM members WHERE name LIKE 'n%';
```

8) Retrieve members sorted by salary in descending order:

```
SQL> SELECT * FROM members ORDER BY salary DESC;
```

9. Retrieve members with email addresses containing 'gmail':

```
SQL> SELECT * FROM members WHERE email LIKE '%gmail%';
```

10. Retrieve the average salary of all members:

```
SQL> SELECT AVG(salary) AS average_salary FROM members;
```

DATA CONTROL LANGUAGE

1) Grant SELECT permission on members table to user john:

```
SQL> GRANT SELECT ON members TO nitharshana;
```

2)Revoke SELECT permission on members table from user navis:

```
SQL> REVOKE SELECT ON members FROM navis;
```

3)Grant multiple privileges

```
SQL> GRANT SELECT, INSERT ON members TO jane;
```

4)Revoke all privileges:

```
SQL> REVOKE ALL ON members FROM jane;
```

5)Grant privileges with the option to grant others:

```
SQL> GRANT SELECT ON members TO admin WITH GRANT OPTION;
```

AGGREGATE FUNCTIONS AND SORTING

1) Find the total salary of all members:

```
SQL> SELECT SUM(salary) AS total_salary FROM members;
```

2)Find the average salary:

```
SQL> SELECT AVG(salary) AS average_salary FROM members;
```

3)Find the highest (maximum) salary:

```
SQL> SELECT MAX(salary) AS max_salary FROM members;
```

4)Find the highest (maximum) salary:

```
SQL> SELECT MAX(salary) AS max_salary FROM members;
```

5)Count how many members are in the table:

```
SQL> SELECT COUNT(*) AS total_members FROM members;
```

6)Group members by salary and count how many have the same salary:

```
SQL> SELECT salary, COUNT(*) AS number_of_members

     FROM members

     GROUP BY salary;
```

7)Sort members by salary in ascending order:

```
SQL> SELECT * FROM members ORDER BY salary ASC;
```

8)Sort members by salary in descending order:

```
SQL> SELECT * FROM members ORDER BY salary DESC;
```

9)Sort members alphabetically by name:

```
SQL> SELECT * FROM members ORDER BY name ASC;
```

10)Sort by salary (descending), then name (ascending):

```
SQL> SELECT * FROM members ORDER BY salary DESC, name ASC;
```

11)Absolute Value:

```
SQL> SELECT ABS(20) AS "ABSOLUTE VALUE" FROM DUAL;
```

12) Round a Number:

```
SQL> SELECT ROUND(1738.56) AS "ROUND" FROM DUAL;
```

13) Power Function:

```
SQL> SELECT POWER(3, 2) AS "POWER" FROM DUAL;
```

14)Square Root:

```
SQL> SELECT SQRT(25) AS "SQUARE ROOT" FROM DUAL;
```

15). Exponent:

```
SQL> SELECT EXP(5) AS "EXPONENT" FROM DUAL;
```

16)Extract Month:

```
SQL> SELECT EXTRACT(MONTH FROM SYSDATE) AS "MONTH" FROM DUAL;
```

17)Extract Year from Specific Date:

```
SQL> SELECT EXTRACT(YEAR FROM DATE '2018-07-07') AS "YEAR" FROM DUAL;
```

18) Greatest:

```
SQL> SELECT GREATEST(4, 10, 20) AS "NUMBER" FROM DUAL;
```

19) Least:

```
SQL> SELECT LEAST(4, 10, 20) AS "NUMBER" FROM DUAL;
```

20)Modulo:

```
SQL> SELECT MOD(15, 8) AS "NUMBER" FROM DUAL;
```

21) Truncate:

```
SQL> SELECT TRUNC(138.356, 1) AS "NUMBER" FROM DUAL;
```

22) Floor and Ceil:

```
SQL> SELECT FLOOR(28.6) AS "NUMBER" FROM DUAL;

SQL> SELECT CEIL(38.6) AS "NUMBER" FROM DUAL;
```

23) To Date:

```
SQL> SELECT TO_DATE('04-JUL-2018','DD-MON-YYYY') FROM DUAL;
```

24) LTRIM Example:

```
SQL> SELECT LTRIM('   nitharshana') AS "ModifiedName" FROM DUAL;
```

25)SUBSTR:

```
SQL> SELECT SUBSTR('WELCOME', 3, 2) FROM DUAL;
```

26)ASCII of 'A':

```
SQL> SELECT ASCII('A') FROM DUAL;
```

27) Lowercase names:

```
SQL> SELECT LOWER(name) FROM members;
```

28) INITCAP (First letter capitalized):

```
SQL> SELECT INITCAP(name) FROM members;
```

29)Length:

```
SQL> SELECT LENGTH('GPAY') FROM DUAL;
```

30) Uppercase names:

```
SQL> SELECT UPPER(name) FROM members;
```

31) Group by Salary and Calculate Total Members:

```
SQL> SELECT salary, COUNT(*) AS "NUM_MEMBERS" FROM members GROUP BY
salary;
```

32) Find Names Containing 'a':

```
SQL> SELECT name FROM members WHERE INSTR(name, 'a') > 0;
```

33)Grouping by salary, sorting by count:

```
SQL> SELECT salary, COUNT(*) AS "COUNT"

     FROM members

     GROUP BY salary

     ORDER BY COUNT DESC;
```

SET OPERATIONS

1)Retrieves all columns (*) from the branch table where the location is 'chennai'.

```
SQL> SELECT*FROM branch WHERE location='chennai';


 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch                chennai
```

2)Selects all records from the branch table where the branch_name contains the word 'branch' anywhere in the text, using the LIKE '%branch%' pattern.

```
SQL> SELECT*FROM branch WHERE branch_name LIKE'%branch%';

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch                chennai
         4 south branch                   salem
```

3)Retrieves all records from the branch table where the branch_id is less than or equal to 4.

```
SQL> SELECT*FROM branch WHERE branch_id<=4;

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         1 centeal city library           coimbatore
         2 westside branch              chennai
         3 north campus library         erode
         4 south branch                 salem
```

4) Retrieves all records from the branch table where the location is not equal to 'coimbatore'.

```
SQL> SELECT*FROM branch WHERE location!='coimbatore';

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch              chennai
         3 north campus library         erode
         4 south branch                 salem
```

5) Retrieves all records from the branch table where the branch_id is greater than 2.

```
SQL> SELECT*FROM branch WHERE branch_id>2;

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         3 north campus library         erode
         4 south branch                 salem
```

6) Retrieves all records from the branch table where the branch_id is between 2 and 4 inclusive

```
SQL> SELECT*FROM branch WHERE branch_id BETWEEN 2 AND 4;

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch              chennai
         3 north campus library         erode
         4 south branch                 salem
```

7) Retrieves all records from the branch table where the branch_id is greater than or equal to 1.

```
SQL> SELECT*FROM branch WHERE branch_id>=1;


 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         1 centeal city library         coimbatore
         2 westside branch              chennai
         3 north campus library         erode
         4 south branch                  salem
```

8)Retrieves the record(s) from the branch table where the branch_id is equal to the highest (MAX) branch_id value in the entire table

```
SQL> SELECT*FROM branch WHERE branch_id=(SELECT MAX(branch_id)FROM branch);



 BRANCH_ID BRANCH_NAME                   LOCATION
---------- ----------------------------- -----------------------------
         4 south branch                  salem
```

9)Fetches all records from the branch table where the location is either 'chennai' or 'Madurai'; in this case, only the branch in 'chennai' is found.

```
SQL>SELECT*FROM branch WHERE location IN('chennai','Madurai');

 BRANCH_ID BRANCH_NAME                   LOCATION
---------- ----------------------------- -----------------------------
         2 westside branch                 chennai
```

10) Retrieves all records from the branch table and sorts them in ascending order based on the branch_name column.

```
 SQL> SELECT*FROM branch ORDER BY branch_name ASC;

 BRANCH_ID BRANCH_NAME                   LOCATION
---------- ----------------------------- -----------------------------
         1 centeal city library          coimbatore
         3 north campus library              erode
         4 south branch                  salem
         2 westside branch                 chennai
```

11)All records from the branch table where the location value is NULL , but no such rows were found.

```
SQL> SELECT*FROM branch WHERE location is NULL;
```

```
no rows selected
```

12)Retrieves all records from the branch table where the location is not 'coimbatore' or 'salem', returning only branches located in 'chennai' and 'erode'.

```
SQL> SELECT*FROM branch WHERE location NOT IN('coimbatore','salem');



 BRANCH_ID BRANCH_NAME                  LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch              chennai
         3 north campus library          erode
```

13)Retrieves all records from the branch table where the location is either 'chennai' or 'salem'.

```
SQL> SELECT*FROM branch WHERE location IN('chennai','salem');

 BRANCH_ID BRANCH_NAME                  LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch              chennai
         4 south branch                salem
```

14) Attempts to retrieve all records from the branch table where the branch_name starts with the letter 'M', but no such records exist

```
SQL> SELECT*FROM branch WHERE branch_name LIKE'M%';

no rows selected
```

15) Retrieves all records from the branch table where the location is either 'chennai' or 'erode', returning matching rows from both locations.

```
SQL> SELECT*FROM branch WHERE location='chennai'OR location='erode';

 BRANCH_ID BRANCH_NAME                  LOCATION
---------- ------------------------------ ------------------------------
         2 westside branch              chennai
         3 north campus library          erode
```

16)Calculates the average branch_id for branches located in either 'chennai' or 'coimbatore', and displays the result with the alias avg_id_chennai_coimbatore

```
SQL> SELECT AVG(branch_id)AS avg_id_chennai_coimbatore FROM branch WHERE
location='chennai'OR location='coimbatore';


AVG_ID_CHENNAI_COIMBATORE
-------------------------
                      1.5
```

JOINT

1)It retrieves book details along with librarian details where both belong to the same branch

```
SQL> SELECT  b.BOOK_ID, b.TITLE, b.AUTHOR, b.PUBLISHER,
b.CATEGORYb.AVAILABILITY,l.LIBRARIAN_ID, l.NAME, l.EMAIL, l.PHONE, l.SHIFT_TIME
FROMBOOKS b INNER JOIN LIBRARIANS l ON b.BRANCH_ID = l.BRANCH_ID;

no rows selected
```

2)A LEFT JOIN query that combines data from the BOOKS and LIBRARIANS tables based on matching BRANCH_ID, but also includes books even if there’s no matching librarian.

```
SQL> SELECT 2 b.BOOK_ID, b.TITLE, b.AUTHOR, b.PUBLISHER, b.CATEGORY,
b.AVAILABILITY, b.BRANCH_ID AS BOOK_BRANCH, 3 l.LIBRARIAN_ID, l.NAME, l.EMAIL,
l.PHONE, l.SHIFT_TIME, l.BRANCH_ID AS LIBRARIAN_BRANCH 4 FROM 5 BOOKS b 6
LEFT JOIN 7 LIBRARIANS l ON b.BRANCH_ID = l.BRANCH_ID;
```

3)Retrieves all rows from the LIBRARIANS table, and matches rows from the BOOKS table only if they share the same BRANCH_ID.

```
SQL> SELECT b.BOOK_ID, b.TITLE, b.AUTHOR, b.PUBLISHER, b.CATEGORY,
b.AVAILABILITY, b.BRANCH_ID AS BOOK_BRANCH, l.LIBRARIAN_ID, l.NAME, l.EMAIL,
l.PHONE, l.SHIFT_TIME, l.BRANCH_ID AS   LIBRARIAN_BRANCH FROM BOOKS b RIGHT
JOIN LIBRARIANS l ON b.BRANCH_ID = l.BRANCH_ID;
```

4)A FULL OUTER JOIN between the BOOKS and LIBRARIANS tables, combining all records from both tables based on the common column BRANCH_ID.

```
SQL> SELECT  b.BOOK_ID, b.TITLE, b.AUTHOR, b.PUBLISHER, b.CATEGORY,
b.AVAILABILITY, b.BRANCH_ID AS BOOK_BRANCH, 3 l.LIBRARIAN_ID, l.NAME, l.EMAIL,
l.PHONE, l.SHIFT_TIME, l.BRANCH_ID AS LIBRARIAN_BRANCH 4 FROM 5 BOOKS b 6
FULL OUTER JOIN 7 LIBRARIANS l ON b.BRANCH_ID = l.BRANCH_ID;
```

5)Retrieves records from both the BOOKS and LIBRARIANS tables where the BRANCH_ID matches in both tables

```
SQL> SELECT b.BOOK_ID, b.TITLE, b.AUTHOR, b.PUBLISHER, b.CATEGORY,
b.AVAILABILITY,l.LIBRARIAN_ID, l.NAME, l.EMAIL, l.PHONE, l.SHIFT_TIME FROM BOOKS b
INNER JOIN LIBRARIANS l ON b.BRANCH_ID = l.BRANCH_ID;
```

VIEWS

1)View to Show Members with Non-Empty Phone Numbers

```
SQL> CREATE VIEW MembersWithPhone AS
SELECT name, phone
FROM Member
WHERE phone IS NOT NULL;
```

2)View to Display Members with Email and Address

```
SQL> CREATE VIEW MemberEmailAddress AS
SELECT name, email, address
FROM Member;
```

3)View to Show Members Whose Email is Not Gmail

```
SQL> CREATE VIEW MembersNonGmail AS
SELECT name, email
FROM Member
WHERE email NOT LIKE '%@gmail.com';
```

4)View to Display Members with Long Names

```
SQL>CREATE VIEW MembersLongNames AS
SELECT name
FROM members
WHERE LENGTH(name) > 10;
```

5)The view BranchDetails will show branch_id, branch_name, and location columns from the branch table.

```
SQL>CREATE VIEW BranchDetails AS
SELECT branch_id, branch_name, location
FROM branch;
```

INTEGRITY CONSTRAINTS:

1)SQL>select*from members;

```
 MEMBER_ID NAME                 EMAIL                          SALARY
---------- -------------------- ------------------------------ ----------
    PHONE
----------
         1 nitharshana          nithu@gmail.com                  60000
9876543210

         2 navis evangelin      navis@gmail.com                  50000
8765432109

         3 nidharshini          nidharshini@gmail.com            40000
7654321098


 MEMBER_ID NAME                 EMAIL                          SALARY
---------- -------------------- ------------------------------ ----------
    PHONE
----------
         4 nirupa               nirupa@gmail.com                 50000
6543210987
```

2)SQL> select*from LIBRARIANS;

```
LIBRARIAN_ NAME
---------- --------------------------------------------------
EMAIL                                              PHONE
-------------------------------------------------- ---------------
SHIFT_TIME           BRANCH_ID
-------------------- ----------
L001       NIRUPA
nirupa@example.com                                 9876543210
Morning              B001
```

```
L002       NIDHARSHANA
nidharshana@example.com                         9123456780
Evening             B002

LIBRARIAN_ NAME
---------- ------------------------------------------------
EMAIL                                        PHONE
------------------------------------------------ ---------------
SHIFT_TIME          BRANCH_ID
-------------------- ----------

L003       NIDHARSHINI
nidharshini@example.com                         9012345678
Morning             B001

L004       NAVIS EVANGELIN
navis.evangelin@example.com                     9988776655

LIBRARIAN_ NAME
---------- ------------------------------------------------
EMAIL                                        PHONE
------------------------------------------------ ---------------
SHIFT_TIME          BRANCH_ID
-------------------- ----------
Night               B003

SQL>select*from transaction_table;

TRANSACTION_ID    BOOK_ID  MEMBER_ID   STAFF_ID SELL_DATE
-------------- ---------- ---------- ---------- --------------------
DUE_DATE                       SELLING_PRICE
------------------------------ -------------
           101       1001      10001     100001 2025-01-10
2025-01-15                               150

           102       1002      10002     100002 2025-02-02
2025-02-23                               200

3)SQL> select*from branch;

 BRANCH_ID BRANCH_NAME                    LOCATION
---------- ------------------------------ ------------------------------
         1 centeal city library           coimbatore
         2 westside branch                 chennai
         3 north campus library          erode
         4 south branch                  salem
```

```
4)SQL> select*from issue_return;

TRANSACTION_ID    BOOK_ID  MEMBER_ID LIBRARIAN_ID ISSUE_DAT RETURN_DA
DUE_DATE
-------------- ---------- ---------- ------------ --------- --------- ---------
FINE_DECIMAL
------------
           101       1001      10001       100001 01-JAN-25 10-JAN-25 15-JAN-25
          15

           102       1002      10002       100002 02-FEB-25 02-FEB-25 23-FEB-25
          20
```

PL/SQL;

```
1)PL for issue return in 101
SQL> DECLARE
  2      issue_rec issue_return%ROWTYPE;BEGIN
  3      -- Select the record for transaction_id = 101 into the record variable
  4      SELECT * INTO issue_rec
  5      FROM issue_return
  6      WHERE transaction_id = 101;
  7
  9      DBMS_OUTPUT.PUT_LINE('Transaction ID : ' || issue_rec.transaction_id);
 10      DBMS_OUTPUT.PUT_LINE('Book ID        : ' || issue_rec.book_id);
 11      DBMS_OUTPUT.PUT_LINE('Member ID      : ' || issue_rec.member_id);
 12      DBMS_OUTPUT.PUT_LINE('Librarian ID   : ' || issue_rec.librarian_id);
 13      DBMS_OUTPUT.PUT_LINE('Issue Date     : ' || TO_CHAR(issue_rec.issue_date,
'DD-MON-YYYY'));
 14      DBMS_OUTPUT.PUT_LINE('Return Date    : ' ||
 15                          NVL(TO_CHAR(issue_rec.return_date, 'DD-MON-YYYY'), 'Not Returned'));
 16      DBMS_OUTPUT.PUT_LINE('Due Date       : ' || TO_CHAR(issue_rec.due_date,
'DD-MON-YYYY'));
 18      DBMS_OUTPUT.PUT_LINE('Fine           : ' || issue_rec.fine_decimal);
 19  END;
20  set serveroutput on;
21 /
Transaction ID : 101
Book ID        : 1001
Member ID      : 10001
Librarian ID   : 100001
```

```
Issue Date     : 01-JAN-2025
Return Date    : 10-JAN-2025
Due Date       : 15-JAN-2025
Fine           : 15

PL/SQL procedure successfully completed.
```

2) PL for issue_return in 102

```
SQL> DECLARE
     issue_rec issue_return%ROWTYPE;  -- Declare a variable of ROWTYPE to hold one row
 BEGIN
    SELECT * INTO issue_rec
     FROM issue_return
     WHERE transaction_id = 102;
     DBMS_OUTPUT.PUT_LINE('Transaction ID : ' || issue_rec.transaction_id);
    DBMS_OUTPUT.PUT_LINE('Book ID        : ' || issue_rec.book_id);
   DBMS_OUTPUT.PUT_LINE('Member ID      : ' || issue_rec.member_id);
    DBMS_OUTPUT.PUT_LINE('Librarian ID   : ' || issue_rec.librarian_id);
      DBMS_OUTPUT.PUT_LINE('Issue Date     : ' || TO_CHAR(issue_rec.issue_date, 'DD-MON-YYYY'));
      DBMS_OUTPUT.PUT_LINE('Return Date    : ' ||
          NVL(TO_CHAR(issue_rec.return_date, 'DD-MON-YYYY'), 'Not Returned'));
      DBMS_OUTPUT.PUT_LINE('Due Date       : ' || TO_CHAR(issue_rec.due_date, 'DD-MON-YYYY'));
      DBMS_OUTPUT.PUT_LINE('Fine           : ' || issue_rec.fine_decimal);
  END;
set serveroutput on;
  /
Transaction ID : 102
Book ID        : 1002
Member ID      : 10002
Librarian ID   : 100002
Issue Date     : 02-FEB-2025
Return Date    : 02-FEB-2025
Due Date       : 23-FEB-2025
Fine           : 20

PL/SQL procedure successfully completed.
```

3)PL for BOOKS

```
SQL> SET SERVEROUTPUT ON;
SQL> DECLARE
```

```
    book_rec BOOKS%ROWTYPE;  -- Use the correct table name
BEGIN
    SELECT * INTO book_rec
    FROM BOOKS
    WHERE BOOK_ID = '002';

    DBMS_OUTPUT.PUT_LINE('Book ID      : ' || book_rec.BOOK_ID);
    DBMS_OUTPUT.PUT_LINE('Title        : ' || book_rec.TITLE);
    DBMS_OUTPUT.PUT_LINE('Author       : ' || book_rec.AUTHOR);
    DBMS_OUTPUT.PUT_LINE('Publisher    : ' || book_rec.PUBLISHER);
    DBMS_OUTPUT.PUT_LINE('Category     : ' || book_rec.CATEGORY);
    DBMS_OUTPUT.PUT_LINE('Availability : ' || book_rec.AVAILABILITY);
    DBMS_OUTPUT.PUT_LINE('Branch ID    : ' || book_rec.BRANCH_ID);
END;
/
Book ID      : 002
Title        : IKIGAI
Author       : SRIYOSA
Publisher    : AJ PUBLICATIONS
Category     : PSYCHOLOGY
Availability : 10
Branch ID    : 21345

PL/SQL procedure successfully completed.
```