DATABASE APPLICATION LAB

Structured Enquiry

"LIBRARY MANAGEMENT SYSTEM"

Under the guidance of

Savita Bakare mam

- **Team no: 01**
- **Sem: 3 rd.**
- **Div: B**
- **Department: Computer Science**

Team Details

Name	*SRN*	*ROLL NO*
Abhilash B	O2FE22BCS004	01
Aditi J	O2FE22BCS006	02
. Akhilesh J	O2FE22BCS013	03

Problem Statement:

A to city library in Belagavi have planned to automate it's library operation and member operations, The library lends books, CD's and DVD'S to its members. Library administrator should be to add a book / CD / DVD with book/CD/DVD name, author name, category (Fiction, Entertainment Technical and Management) and number of copies. Member should be registered at the library before he/she is allowed to check out Book/ CD/DVD. He/she needs to be specified for any transactions to be done in the library.

Search facility needs to be made available to members by book name or author name or category or combination of all the mentioned to search the relevant resources. the member chooses one or more book/ CD/DVD to check out. Software needs to ensure the availability of the book before check out. Due date needs to be displayed and once checked out due dates are 10 days ,5 days and 3 days for book, CD and DVD Respectively The member should also be able to see the list of books that he/she checked out when the member returns the books beyond the due date, he / she should be fined accordingly. Fines for book, CD and DVD are Rs 10, Rs 20 and Rs 30 respectively. The administrator should be able to see the list of books checked out or due or returned for a given date

Objectives

- Efficient Information Organization: Ensure systematic arrangement and retrieval of library
- Resources using a well-designed database structure.User Accessibility: Facilitate easy access for users to locate and borrowbooks,reducing the time and effort required to find relevant materials.
- User Management:Maintain User Information,track borrowing history, and provide functionalities for user registration,renewal,and account updates.

- User-FriendlyInterface:Develop an intuitive interface for both library staff and users,enhancing the overall user experience and making it easier to navigate the system.

Assumptions / Preliminary ideas followed

- User-FriendlyInterface:
- OnlineCatalogandSearchFunctionality:
- UserAccountManagement:
- FineManagementSystem

Feasibility study

Existing Issues:

- Manual Cataloging
- Limited Accessibility
- Ineffective Tracking

Needs for a New Library Management System:

- Efficient Check-Out/Check-In
- Automated Overdue Item Tracking:
- Resources Availability
- Servers,computers,and   networking infrastructure.
- Library Management System software.
- Stafftrainingonthenewsystem.

Requirement Analysis

- Enable user registration with personal details, account profile management ,and the ability to track user activities,such as borrowing history.
- An automated fine management system with calculations for overdue items, notifications,and a secure fine payment system.

Objective

The objective of automating the library operations in Belagavi is to enhance efficiency, accuracy, and accessibility in managing library resources and member transactions. The automation system aims to streamline tasks such as book/CD/DVD management, member registration, and transaction tracking. It intends to provide an efficient search mechanism for members and timely information for administrators. The automation system also focuses on reducing manual efforts, minimizing errors, and improving the overall library experience

Scope

The scope of the library automation system includes the implementation of features like adding and managing books, CDs, and DVDs, member registration, and transaction tracking. It encompasses functionalities such as searching for resources by various criteria, checking out items, displaying due dates, managing returns, and calculating fines for overdue items. The system aims to provide both members and administrators with user-friendly interfaces to carry out their respective tasks effectively. The scope also covers generating reports for administrators to monitor library activities.

User Requirements

1.Library Administrators:

- Add books, CDs, and DVDs with details (name, author, category, copies available).
- Register new members with personal information.
- View and manage transactions (checkouts, returns) for monitoring library activities.
- Generate reports on checked-out items, due dates, and fines.

2.Library Members:

- Search for resources by book/author/category combinations.
- View availability of books, CDs, and DVDs.
- Check out items and receive due dates.
- View a list of checked-out items.
- Return items on or before the due date to avoid fines.

NORMALISATION

NORMALISATION:

R (book_name ,author_name, category, no_of_copies, b_mid, b_issue_date,b_due_date,b_return_date, name, category, no_of_copies, c_mid, c_issue_date, c_due_date, c_return_date, _name, category, no_of_copies, d_mid, d_issue_date, d_due_date,d_return_date,fine_id,type,amount**)**

FUNCTIONAL DEPENDENCIES:

book_id→ book_name ,author_name, category, no_of_copies, b_mid, b_issue_date,b_due_date,b_return_date.

cd_id→ cd_name, category, no_of_copies, c_mid, c_issue_date, c_due_date, c_return_date.

dvd_id→dvd_name, category, no_of_copies, d_mid, d_issue_date, d_due_date,d_return_date.

fine_id→type,amount

FIRST NORMAL FORM:

Since the data values are atomic,the tables are in first normal form.

SECOND NORMAL FORM:

STEP 1: Find the candidate key

Candidate key→(book_id, cd_id, dvd_id, b_mid, c_mid, d_mid)

(book_id)*→ book_name ,author_name, category, no_of_copies

(cd_id)*→ cd_name, category, no_of_copies

(dvd_id)*→ dvd_name, category, no_of_copies

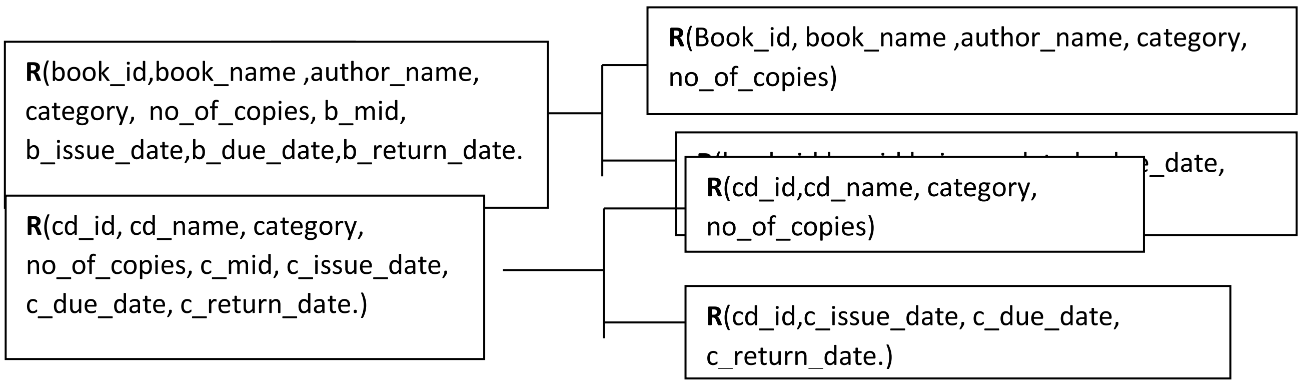
(b_mid)*→ b_mid, b_issue_date,b_due_date,b_return_date.

(c_mid)*→ _issue_date, c_due_date, c_return_date.

(dvd_id)*→ d_issue_date, d_due_date,d_return_date.

So,now the above functional dependencies are in second normal form.

The decomposed tables are:



R(book_id,book_name ,author_name, category, no_of_copies, b_mid, b_issue_date,b_due_date,b_return_date.

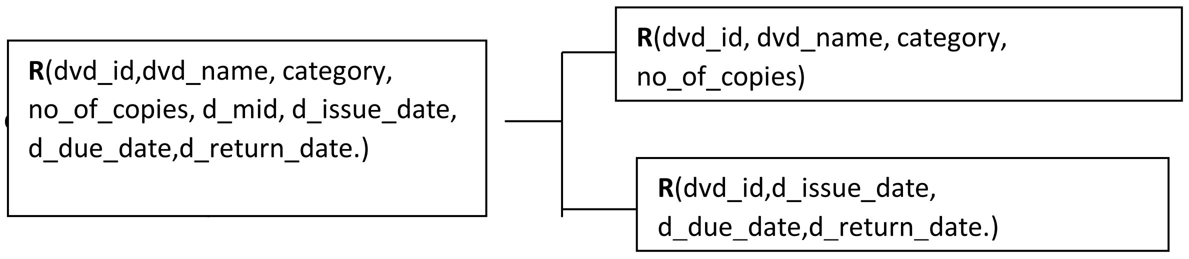
R(Book_id, book_name ,author_name, category, no_of_copies)

[illegible] _date,

R(cd_id, cd_name, category, no_of_copies, c_mid, c_issue_date, c_due_date, c_return_date.)

R(cd_id,cd_name, category, no_of_copies)

R(cd_id,c_issue_date, c_due_date, c_return_date.)


R(dvd_id,dvd_name, category, no_of_copies, d_mid, d_issue_date, d_due_date,d_return_date.)
R(dvd_id, dvd_name, category, no_of_copies)
R(dvd_id,d_issue_date, d_due_date,d_return_date.)

THIRD NORMAL FORM AND BCNF:

Since the left hand side of the functional dependencies is a single attribute,it is already in third

.

.

.

..

.

.

.

.

.

.

.

.

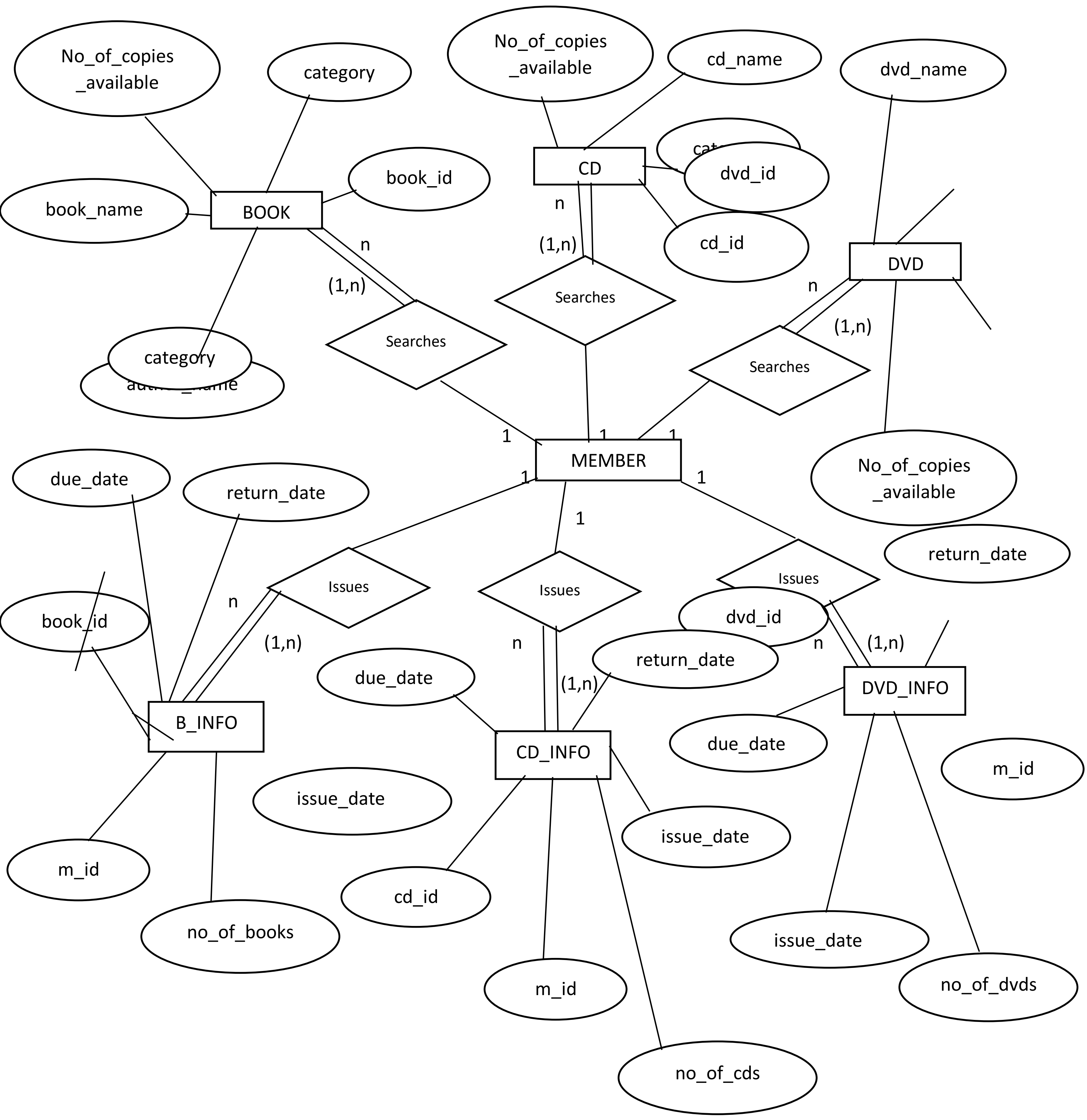
.

.

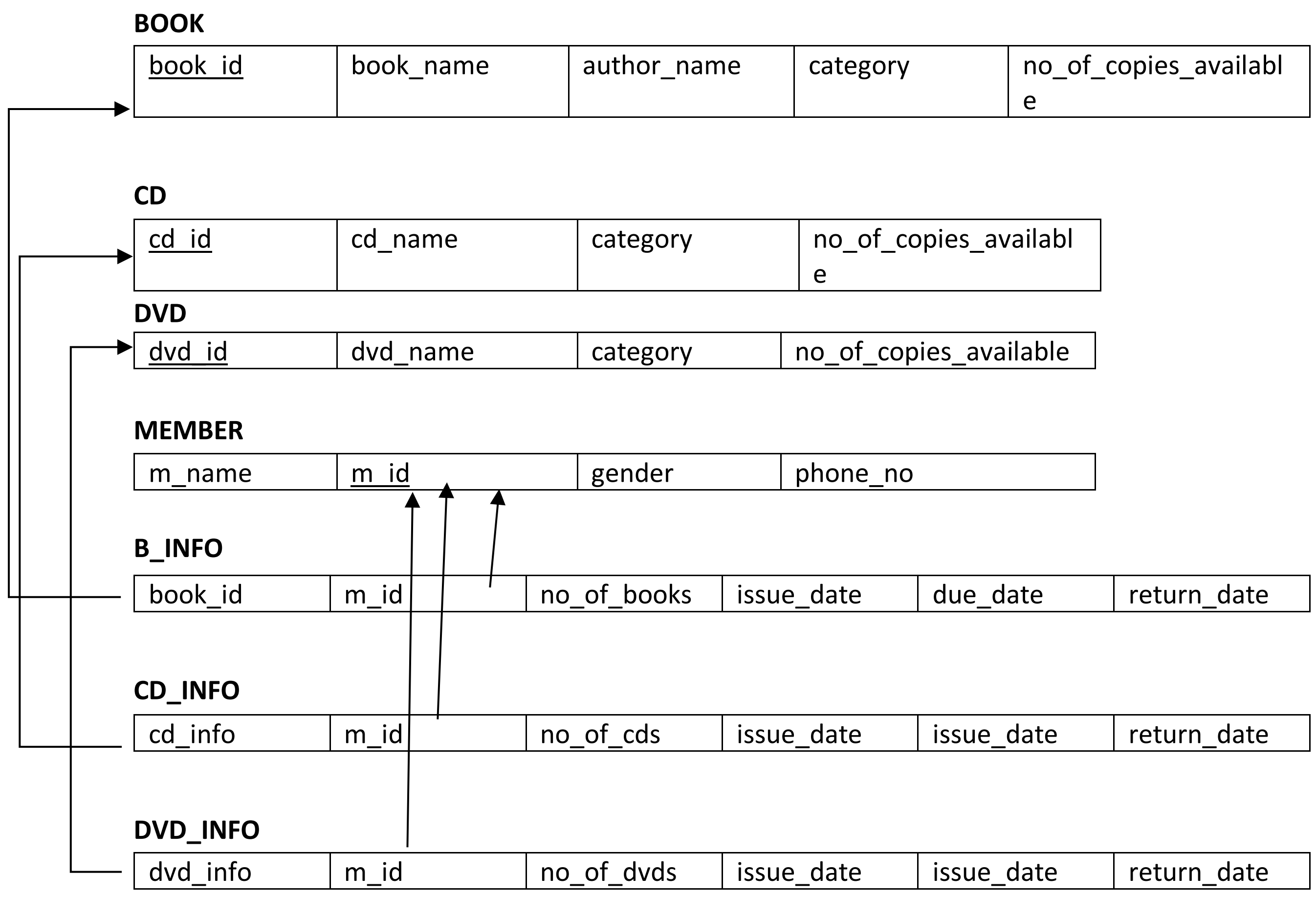
.

P.T.O

E-R MODEL [Entity Relationship Model



Relational Schema



BOOK

<u>book_id</u>	book_name	author_name	category	no_of_copies_availabl e

CD

<u>cd_id</u>	cd_name	category	no_of_copies_availabl e

DVD

<u>dvd_id</u>	dvd_name	category	no_of_copies_available

MEMBER

m_name	<u>m_id</u>	gender	phone_no

B_INFO

book_id	m_id	no_of_books	issue_date	due_date	return_date

CD_INFO

cd_info	m_id	no_of_cds	issue_date	issue_date	return_date

DVD_INFO

dvd_info	m_id	no_of_dvds	issue_date	issue_date	return_date

Primary and Foreign Keys

Primary keys

<u>book_id</u>	in book table
<u>cd_id</u>	in cd table
<u>dvd_id</u>	in dvd table
<u>m_id</u>	In member table

Foreign keys

book_id and m_id in b_info table

cd _id and m_id in cd_info table

dvd_id and m_id in dvd_id table

Inserted Data

BOOK

BOOK_ID	BOOK_NAME	AUTHOR_NAME	CATEGORY	COPIES_AVAIL
1	Haunted	John	Horror	8
2	Impossible	Harry	Mystery	4
3	Young_me	David	Biography	5
4	Sunrise	Selena	Poetry	8
5	Ghost	John	Horror	3
6	Day_out	Richard	Entertainment	6
7	Reflection	Diana	Biography	2
8	Sunset	Selena	Poetry	8
9	Laugh	Stephen	Entertainment	7
10	The_house	Harry	Mystery	4

Cd

CD_ID	CD_NAME	CATEGORY	COPIES_AVAIL
1	Echoes	Horror	4
2	Colors	Entertainment	5
3	The_Journey	Poetry	6
4	Mansion	Stories	4
5	Fight	Entertainment	4
6	Dark	Horror	8
7	Moonlight	Poetry	1
8	Good_lady	Stories	4

DVD

DVD_ID	DVD_NAME	CATEGORY	COPIES_AVAIL
1	Robot	Adventure	4
2	Commander	Adventure	5
3	Seabiscuit	Drama	3
4	Weapon	Drama	6
5	Life	Adventure	7
6	Boat_trip	Comedy	4

MEMBER

M_ID	M_NAME	G	PHONE_NO
1	Ravi	m	3452673677
2	Sakshi	f	3452672288
3	Sujal	m	3657873622
4	Neeta	f	3452673644
5	Maya	f	3452673679
6	Rahul	m	3452673693
7	Priya	f	3452673699
8	Simran	f	3452673375
9	Roshan	m	3452673690
10	Shiv	m	3452672347

B_info

BOOK_ID	M_ID	NO_OF_BOKOS	DUE_DATE	DATE OF RETURN	RETURN_DATE
3	2	2	01-JAN-23	10-JAN-23	12-JAN-23
1	4	3	04-JAN-23	14-JAN-23	14-JAN-23
6	1	1	07-JAN-23	17-JAN-23	18-JAN-23
10	6	2	18-FEB-23	28-FEB-23	26-FEB-23
2	4	2	02-MAR-23	12-MAR-23	12-MAR-23
6	8	1	08-JUN-23	18-JUN-23	19-JUN-23
5	3	1	17-JUL-23	27-JUL-23	28-JAN-23
2	3	1	19-AUG-23	29-AUG-23	29-AUG-23

Cd_info

CD_ID	M_ID	NO_OF_BOOKS	ISSUE_DATE	DUE_DATE	RETURN_DATE
2	2	1	07-JAN-23	12-JAN-23	13-JAN-23
3	1	2	04-SEP-23	09-SEP-23	09-SEP-23
5	6	1	04-MAR-23	09-MAR-23	10-MAR-23
7	2	2	14-SEP-23	19-SEP-23	19-SEP-23
6	2	1	17-APR-23	27-APR-23	24-APR-23
5	1	1	01-MAY-23	06-MAY-23	08-MAY-23
8	3	2	03-SEP-23	08-SEP-23	08-SEP-23
4	4	2	08-DEC-23	18-DEC-23	28-DEC-23
1	9	1	05-MAR-23	08-MAR-23	10-MAR-23
3	3	2	04-APR-23	07-APR-23	07-APR-23
2	4	1	12-AUG-23	15-AUG-23	15-AUG-23

Queries

JOIN QUERIES

1.Retrieve member names and their borrowed book information:

```
SELECT m.m_name, b.book_name, bi.issue_date, bi.due_date, bi.return_date
FROM member m
JOIN b_info bi ON m.m_id = bi.m_id
JOIN book b ON b.book_id = bi.book_id;
```

OUTPUT

```
M_NAME              BOOK_NAME           ISSUE_DAT DUE_DATE  RETURN_DA
-------------------- -------------------- --------- --------- ---------
Neeta               Haunted             04-JAN-23 14-JAN-23 14-JAN-23
Neeta               Impossible          02-MAR-23 12-MAR-23 12-MAR-23
Sujal               Impossible          19-AUG-23 29-AUG-23 29-AUG-23
Sakshi              Young_me            01-JAN-23 10-JAN-23 12-JAN-23
Sujal               Ghost               17-JUL-23 27-JUL-23 28-JAN-23
Simran              Day_out             08-JUN-23 18-JUN-23 19-JUN-23
Ravi                Day_out             07-JAN-23 17-JAN-23 18-JAN-23
Rahul               The_house           18-FEB-23 28-FEB-23 26-FEB-23
```

2.Retrieve books borrowed by male members:

```
SELECT m.m_name, b.book_name
FROM member m
JOIN b_info bi ON m.m_id = bi.m_id
JOIN book b ON b.book_id = bi.book_id
WHERE m.gender = 'm';
```

OUTPUT

```
M_NAME              BOOK_NAME
-------------------- --------------------
Sujal               Impossible
Sujal          Ghost
Ravi           Day_out
Rahul            The_house
```

3.Retrieve the names of books returned before their due date:

```
  SELECT b.book_name
  FROM book b
  JOIN b_info bi ON b.book_id = bi.book_id
  WHERE bi.return_date < bi.due_date;
```

OUTPUT

```
BOOK_NAME
--------------------
Ghost
The_house
```

4.Retrieve members who borrowed DVDs before '2023-06-01':

```
  SELECT m.m_id, m.m_name
  FROM member m
  JOIN dvd_info di ON m.m_id = di.m_id
  WHERE di.issue_date < DATE '2023-06-01';
```

OUTPUT

```
     M_ID M_NAME
---------- --------------------
         3 Sujal
         3 Sujal
         4 Neeta
         6 Rahul
         7 Priya
         9 Roshan
```

5. Retrieve information about books that have been borrowed, along with member details:

```
SELECT m.m_name, b.book_name, bi.issue_date, bi.due_date, bi.return_date

FROM member m

JOIN b_info bi ON m.m_id = bi.m_id

JOIN book b ON b.book_id = bi.book_id;
```

OUTPUT

```
M_NAME           BOOK_NAME         ISSUE_DAT DUE_DATE  RETURN_DA

-------------------- -------------------- --------- --------- ---------

Neeta            Haunted           04-JAN-23 14-JAN-23 14-JAN-23

Neeta            Impossible        02-MAR-23 12-MAR-23 12-MAR-23

Sujal           Impossible        19-AUG-23 29-AUG-23 29-AUG-23

Sakshi           Young_me          01-JAN-23 10-JAN-23 12-JAN-23

Sujal           Ghost            17-JUL-23 27-JUL-23 28-JAN-23

Simran           Day_out           08-JUN-23 18-JUN-23 19-JUN-23

Ravi            Day_out          07-JAN-23 17-JAN-23 18-JAN-23

Rahul            The_house         18-FEB-23 28-FEB-23 26-FEB-23
```

DIFFERENT CLAUSES & FUNCTIONS

1. Retrieve the total number of copies available for each category of CDs:

```
  SELECT category, SUM(copies_avail) AS total_copies_available

  FROM cd

  GROUP BY category;
```

OUTPUT

```
CATEGORY          TOTAL_COPIES_AVAILABLE

-------------------- ----------------------

Entertainment                  9

Poetry                      7

Horror                     12

Stories                     8
```

2. Find the member who has the maximum number of borrowed items (books, CDs, DVDs combined):

```
 SELECT *
   FROM (
      SELECT m.m_name, COUNT(*) AS total_borrowed_items
       FROM member m
       LEFT JOIN b_info bi ON m.m_id = bi.m_id
       LEFT JOIN cd_info ci ON m.m_id = ci.m_id
      LEFT JOIN dvd_info di ON m.m_id = di.m_id
      GROUP BY m.m_name
      ORDER BY total_borrowed_items DESC
      )
  WHERE ROWNUM <= 1;
```

OUTPUT

```
M_NAME               TOTAL_BORROWED_ITEMS
-------------------- --------------------
Sujal                                   4
```

3. Retrieve the list of DVDs that have not been checked out:

```
  SELECT d.dvd_name
  FROM dvd d
  LEFT JOIN dvd_info di ON d.dvd_id = di.dvd_id
  WHERE di.m_id IS NULL;
```

OUTPUT

4. Calculate the average fine amount for overdue CDs:

```
  SELECT AVG((TO_DATE(ci.return_date, 'DD-MON-YY') - TO_DATE(ci.due_date, 'DD-MON-YY')) * 20)
AS avg_cd_fine
  FROM cd_info ci
  WHERE ci.return_date IS NOT NULL AND ci.return_date > ci.due_date;
```

OUTPUT

```
AVG_CD_FINE
----------
```

5. Retrieve the members who have not borrowed any books, CDs, or DVDs:

```
  SELECT m.m_id, m.m_name
  FROM member m
  LEFT JOIN b_info bi ON m.m_id = bi.m_id
  LEFT JOIN cd_info ci ON m.m_id = ci.m_id
  LEFT JOIN dvd_info di ON m.m_id = di.m_id
  WHERE bi.m_id IS NULL AND ci.m_id IS NULL AND di.m_id IS NULL;
```

OUTPUT

```
    M_ID M_NAME
---------- --------------------
       5 Maya
      10 Shiv
```

These queries involve various SQL clauses such as GROUP BY, ORDER BY, LEFT JOIN, WHERE, and functions like SUM, COUNT, AVG, and LIMIT. .

SUB-QUERIES (single row , Multiple row and co related nested query)

Single-Row Sub-queries:

1. Retrieve the member names who have checked out the most number of books:

```
  SELECT m_name
  FROM member
  WHERE m_id = (SELECT m_id FROM b_info GROUP BY m_id ORDER BY COUNT(*) DESC    LIMIT
1);
```

OUTPUT

2. Find the book with the highest number of available copies:

```
   SELECT book_name
   FROM book
   WHERE copies_avail = (SELECT MAX(copies_avail) FROM book);
```

OUTPUT

```
BOOK_NAME
--------------------
Haunted
Sunrise
Sunset
```

3. Retrieve the due date of the last checked-out CD:

```
   SELECT MAX(due_date)
   FROM cd_info;
```

OUTPUT

```
MAX(DUE_D
---------
```

Multiple-Row Sub-queries:

4. Retrieve members who have borrowed both a book and a DVD:

```
   SELECT m_id, m_name
   FROM member
   WHERE m_id IN (SELECT m_id FROM b_info) AND m_id IN (SELECT m_id FROM dvd_info);
```

OUTPUT

```
      M_ID M_NAME
---------- --------------------
         4 Neeta
         6 Rahul
         8 Simran
         3 Sujal
```

5. Find CDs with more than the average number of copies available:

```
  SELECT cd_name
  FROM cd
  WHERE copies_avail > (SELECT AVG(copies_avail) FROM cd);
```

OUTPUT

```
CD_NAME
--------------------
Colors
The_Journey
Dark
```

6. Retrieve members who have not borrowed any books:

```
  SELECT m_id, m_name
  FROM member
  WHERE m_id NOT IN (SELECT m_id FROM b_info);
```

OUTPUT

Correlated Nested Queries:

7. Retrieve books that have fewer copies available than the average copies for their category;

```
  SELECT book_name, category, copies_avail
  FROM book b
  WHERE copies_avail < (SELECT AVG(copies_avail) FROM book WHERE category = b.category);
```

OUTPUT

```
BOOK_NAME            CATEGORY             COPIES_AVAIL
-------------------- -------------------- ------------
Ghost                Horror                         3
Day_out              Entertainment                  6
Reflection           Biography                      2
```

8. Find members who have borrowed more books than the average number of books borrowed by male members:

```
  SELECT m_id, m_name
  FROM member m
  WHERE (SELECT COUNT(*) FROM b_info WHERE m_id = m.m_id) > (SELECT AVG(COUNT(*)) FROM b_info WHERE m_id IN (SELECT m_id FROM member WHERE gender = 'm') GROUP BY m_id);
```

OUTPUT

```
      M_ID M_NAME
---------- --------------------
         3 Sujal
         4 Neeta
```

9. Retrieve DVDs that have been borrowed by female members in the last month:

```
  SELECT dvd_name
  FROM dvd d
  WHERE EXISTS (SELECT 1 FROM dvd_info di WHERE d.dvd_id = di.dvd_id AND di.issue_date >= SYSDATE - INTERVAL '1' MONTH AND di.m_id IN (SELECT m_id FROM member WHERE gender = 'f'));
```

OUTPUT

Mixed Sub-queries:

10. Retrieve members who have borrowed both a CD and a DVD:

```
SELECT m_id, m_name
FROM member
WHERE m_id IN (SELECT m_id FROM cd_info) AND m_id IN (SELECT m_id FROM dvd_info);
```

OUTPUT

11. Find books that have been borrowed more than once:

```
SELECT book_name
FROM book
WHERE book_id IN (SELECT book_id FROM b_info GROUP BY book_id HAVING COUNT(*) > 1);
```

OUTPUT

```
BOOK_NAME
--------------------
Day_out
Impossible
```

12. Retrieve members who have not returned any borrowed items:

```
SELECT m_id, m_name
FROM member
WHERE m_id NOT IN (SELECT m_id FROM b_info WHERE return_date IS NOT NULL
                   UNION ALL
                   SELECT m_id FROM cd_info WHERE return_date IS NOT NULL
                   UNION ALL
                   SELECT m_id FROM dvd_info WHERE return_date IS NOT NULL);
```

OUTPUT

13. Retrieve the names of members who have borrowed the same book more than once:

```
SELECT m_name
```

```
    FROM member
    WHERE m_id IN (SELECT m_id FROM b_info WHERE book_id IN (SELECT book_id FROM b_info
GROUP BY book_id HAVING COUNT(*) > 1));
```

OUTPUT

14. Find the CDs with the most number of copies borrowed by any member:

```
    SELECT cd_name
    FROM cd
    WHERE cd_id IN (SELECT cd_id FROM cd_info GROUP BY cd_id ORDER BY COUNT(*) DESC
LIMIT 1);
```

OUTPUT

15. Retrieve members who have borrowed a DVD after returning a book in the same category:

```
    SELECT m_id, m_name
    FROM member
    WHERE EXISTS (
        SELECT 1
        FROM b_info bi
        JOIN book b ON bi.book_id = b.book_id
        WHERE bi.m_id = member.m_id
        AND EXISTS (
            SELECT 1
            FROM dvd_info di
            JOIN dvd d ON di.dvd_id = d.dvd_id
            WHERE di.m_id = member.m_id
            AND b.category = d.category
            AND di.issue_date > bi.return_date
        )
    );
```

These queries cover a variety of scenarios using sub-queries, including comparisons, aggregations, and existence checks.

Views

1. Create a view that shows all available books with their details:

```
CREATE VIEW available_books AS

SELECT book_id, book_name, author_name, category, copies_avail

FROM book

WHERE copies_avail > 0;
```

2. Retrieve information about members who have borrowed DVDs using a view:

```
CREATE VIEW members_with_dvds AS

SELECT m.*

FROM member m

INNER JOIN dvd_info di ON m.m_id = di.m_id;
```

3. Create a view that displays the total count of CDs in each category:

```
CREATE VIEW cd_category_counts AS

SELECT category, COUNT(*) AS cd_count

FROM cd

GROUP BY category;
```

4. Create a view that displays the list of members who have borrowed books along with due dates:

```
CREATE VIEW member_borrowed_due_dates AS
SELECT m.m_id, m.m_name, b.book_name, bi.due_date
FROM member m
JOIN b_info bi ON m.m_id = bi.m_id
JOIN book b ON b.book_id = bi.book_id;
```

OUTPUT

5. Retrieve information about CDs that have more than 3 available copies using a view:

```
CREATE VIEW cds_with_many_copies AS
SELECT cd_id, cd_name, category, copies_avail
FROM cd
WHERE copies_avail > 3;
```