

AI ASSISTED CODING

NAME: KASU SIRI CHANDANA

ROLL NO :2403A51282

BATCH : 12

ASSIGNMENT : 16

#TASK-1

PROMPT :

Design a schema for a Library Management System
(Tables: Books, Members, Loans).

CODE :

```
SQL*Plus: Release 11.2.0.2.0 Production on Wed Oct 29 08:37:16 2025  
Copyright (c) 1982, 2014, Oracle. All rights reserved.  
  
SQL> connect  
Enter user-name: system  
Enter password:  
Connected.  
SQL> CREATE TABLE Members (  
2     member_id INT PRIMARY KEY,  
3     name VARCHAR(100),  
4     email VARCHAR(100) UNIQUE,  
5     join_date DATE  
6 );  
  
Table created.
```

```
SQL> CREATE TABLE Books (  
2     book_id INT PRIMARY KEY,  
3     title VARCHAR2(200),  
4     author VARCHAR2(100),  
5     available CHAR(1) CHECK (available IN ('Y', 'N'))  
6 );  
  
Table created.
```

```
SQL> CREATE TABLE Loans (  
2     loan_id INT PRIMARY KEY,  
3     member_id INT,  
4     book_id INT,  
5     loan_date DATE,  
6     return_date DATE,  
7     FOREIGN KEY (member_id) REFERENCES Members(member_id),  
8     FOREIGN KEY (book_id) REFERENCES Books(book_id)  
9 );  
  
Table created.
```

OBSERVATION :

- ❑ loan_id: Unique ID for each loan transaction.
- ❑ member_id: References the borrowing member.
- ❑ book_id: References the borrowed book.
- ❑ loan_date: The date the book was borrowed.
- ❑ return_date: The date the book is (or should be) returned.
- ❑ **Foreign keys** ensure referential integrity – a loan must be linked to an existing member and book.

#TASK - 2

PROMPT :

Generate INSERT INTO queries for the schema above
(3 sample records per table).

CODE:

```
SQL> INSERT INTO Books (book_id, title, author, available)  
2 VALUES (101, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Y');  
  
1 row created.  
  
SQL>  
SQL> INSERT INTO Books (book_id, title, author, available)  
2 VALUES (102, 'To Kill a Mockingbird', 'Harper Lee', 'Y');  
  
1 row created.  
  
SQL>  
SQL> INSERT INTO Books (book_id, title, author, available)  
2 VALUES (103, '1984', 'George Orwell', 'Y');  
  
1 row created.
```

```

SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (101, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Y');

1 row created.

SQL>
SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (102, 'To Kill a Mockingbird', 'Harper Lee', 'Y');

1 row created.

SQL>
SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (103, '1984', 'George Orwell', 'Y');

1 row created.

SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (104, 'The Great Gatsby', 'F. Scott Fitzgerald', 'Y');

1 row created.

SQL>
SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (105, 'To Kill a Mockingbird', 'Harper Lee', 'Y');

1 row created.

SQL>
SQL> INSERT INTO Books (book_id, title, author, available)
  2  VALUES (106, '1984', 'George Orwell', 'Y');

1 row created.

```

OBSERVATION :

- ☐ Proper table relationship (foreign key integrity) was maintained throughout.
- ☐ Errors provided valuable learning about **constraint violations** and **execution order**.

#TASK -3

PROMPT:

Generate a query to list all books borrowed by a specific member

CODE:

```
SQL> SELECT b.book_id, b.title, b.author, l.loan_date, l.return_date
 2  FROM Books b
 3  JOIN Loans l ON b.book_id = l.book_id
 4  JOIN Members m ON l.member_id = m.member_id
 5  WHERE m.member_id = 1;

no rows selected
```

OBSERVATION :

1. Used **JOIN operations** between Members, Books, and Loans to fetch related data.
2. Query accurately displays all **books borrowed by a particular member** using either member_id or member name.
3. Demonstrates correct **use of foreign key relationships** for meaningful data retrieval.
4. Output confirms the logical link between tables works properly.

#TASK - 4

PROMPT :

Generate queries with AI for:

- Updating a book's availability to FALSE when borrowed.
- Deleting a member record safely.

CODE :

```
SQL> UPDATE Books
 2  SET available = 'N'
 3  WHERE book_id = 101;

1 row updated.

SQL> DELETE FROM Loans
 2  WHERE member_id = 3;

0 rows deleted.

SQL>
SQL> DELETE FROM Members
 2  WHERE member_id = 3;

1 row deleted.
```

OBSERVATION :

1. **Update Query** correctly changes a book's status from 'Y' to 'N' to mark it unavailable.
2. **Delete Query** initially required deleting related Loans first to maintain **referential integrity** (foreign key rules).
3. Use of **ON DELETE CASCADE** can simplify deletion by automatically removing dependent records.
4. Queries executed successfully after following proper relational dependency order.