DATABASE MANAGEMENT SYSTEMS LAB

REPORT ASSIGNMENT-3

NAME: SOHAM LAHIRI

CLASS: BCSE UG-III 6TH SEMESTER

ROLL NO: 002210501107

GROUP: A3

SUBMISSION DATE: 19/03/2025

Problem Statement 1:

In an organization, number of departments exists. Each department has a name and unique code. Number of employees work in each department. Each employee has unique employee code. Detailed information like name, address, city, basic, date of join are also stored. In a leave register for each employee leave records are kept showing leave type (CL/EL/ML etc.), from date, to-date. When an employee retires or resigns then all the leave information pertaining to him are also deleted. Basic salary must be within Rs.50000 to Rs.90000. A department can not be deleted if any employee record refers to it. Valid grades are A/B/C. Employee name must be in uppercase only. Default value for joining date is system date. Design and implement the tables with necessary constraints to support the scenario depicted above.

SQL Commands:

```
CREATE TABLE ORGANIZATION(
ORG_ID CHAR(5) PRIMARY KEY);

CREATE TABLE DEPARTMENT(
DCODE CHAR(5) PRIMARY KEY,
DNAME CHAR(25),
ORG_ID CHAR(5) NOT NULL,
FOREIGN KEY(ORG_ID) REFERENCES ORGANIZATION(ORG_ID));

CREATE TABLE EMPLOYEE(
ECODE CHAR(5) PRIMARY KEY,
ENAME CHAR(25) CHECK(ENAME=UPPER(ENAME)),
ADDRESS CHAR(25),
CITY CHAR(10),
BASIC INT CHECK(BASIC>=50000 AND BASIC<=90000),
JN_DT DATE DEFAULT SYSDATE,
DCODE CHAR(5) NOT NULL,
FOREIGN KEY(DCODE) REFERENCES DEPARTMENT(DCODE));

CREATE TABLE LEAVE_REGISTER(
```

```
LEAVE_TYPE CHAR(5) NOT NULL CHECK(LEAVE_TYPE='CL' OR
LEAVE_TYPE='EL' OR LEAVE_TYPE='ML'),
FROM_DT DATE,
TO_DT DATE,
ECODE CHAR(5) NOT NULL,
FOREIGN KEY(ECODE) REFERENCES EMPLOYEE(ECODE),
PRIMARY KEY(ECODE,LEAVE_TYPE,FROM_DT));

DESC ORGANIZATION;
DESC EMPLOYEE;
DESC LEAVE_REGISTER;
```

Output:

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
ORG_ID                                    NOT NULL CHAR(5)
```

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
ECODE                                     NOT NULL CHAR(5)
ENAME                                              CHAR(25)
ADDRESS                                            CHAR(25)
CITY                                               CHAR(10)
BASIC                                              NUMBER(38)
JN_DT                                              DATE
DCODE                                     NOT NULL CHAR(5)
```

```
Name                                      Null?    Type
----------------------------------------- -------- ----------------------------
LEAVE_TYPE                                NOT NULL CHAR(5)
FROM_DT                                   NOT NULL DATE
TO_DT                                              DATE
ECODE                                     NOT NULL CHAR(5)
```

Problem Statement 2: Try to violate the constraints that you have implemented in the table and note, what happens. [Try with suitable INSERT/UPDATE/DELETE instruction]

1. Primary Key Constraint Violation: Each primary key must be unique.

SQL Commands:

INSERT INTO ORGANIZATION (ORG_ID) VALUES ('O001');

INSERT INTO ORGANIZATION (ORG_ID) VALUES ('O001'); -- Violation: Duplicate ORG_ID

Result:
ORA-00001: unique constraint (PRIMARY KEY) violated

2. Foreign Key Constraint Violation: Foreign key values must exist in the referenced table.

SQL Commands:

INSERT INTO DEPARTMENT (DCODE, DNAME, ORG_ID) VALUES ('D001', 'HR', 'O999'); -- O999 does not exist

Result:
ORA-02291: integrity constraint violated - parent key not found

SQL Commands:

INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E001', 'JOHN DOE', '123 St', 'NY', 60000, 'D999'); -- D999 does not exist

Result:
ORA-02291: integrity constraint violated - parent key not found

3. CHECK Constraint Violation: These constraints enforce conditions on column values.

(a) Uppercase Employee Name

SQL Commands:

INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E002', 'John Doe', '123 St', 'NY', 60000, 'D001'); -- 'John Doe' is not in uppercase

Result:
ORA-02290: check constraint (ENAME=UPPER(ENAME)) violated

(b) Basic Salary Constraint (Min: 50000, Max: 90000)

SQL Commands:

```
INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E003', 'MARK', '456 St', 'LA', 45000, 'D001'); -- 45000 is less than 50000
```

Result:
ORA-02290: check constraint (BASIC>=50000 AND BASIC<=90000) violated

```
INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E004', 'ALICE', '789 St', 'SF', 95000, 'D001'); -- 95000 is greater than 90000
```

SQL Commands:

Result:
ORA-02290: check constraint (BASIC>=50000 AND BASIC<=90000) violated

(c) Leave Type Constraint

SQL Commands:

```
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)

VALUES ('SL', '2024-08-01', '2024-08-05', 'E001'); -- 'SL' is not allowed
```

Result:
ORA-02290: check constraint (LEAVE_TYPE='CL' OR LEAVE_TYPE='EL' OR LEAVE_TYPE='ML') violated

4. Default Value Check: The JN_DT column in EMPLOYEE has a default value of SYSDATE.

SQL Commands:

```
INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E005', 'DAVID', '321 St', 'MI', 55000, 'D001');

SELECT ECODE, JN_DT FROM EMPLOYEE WHERE ECODE = 'E005';
```

Result:
JN_DT should be automatically set to the current system date.

5. Primary Key Constraint Violation in LEAVE_REGISTER: Since (ECODE, LEAVE_TYPE, FROM_DT) is the primary key, inserting duplicate values is not allowed.

SQL Commands:

```
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)

VALUES ('CL', '2024-08-01', '2024-08-05', 'E001');
```

INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)

VALUES ('CL', '2024-08-01', '2024-08-05', 'E001'); -- Duplicate entry

Result:
ORA-00001: unique constraint (PRIMARY KEY) violated

6. Deleting Parent Record (Foreign Key Violation): If an ORG_ID in ORGANIZATION is referenced in DEPARTMENT, deleting it should fail.

SQL Commands:

DELETE FROM ORGANIZATION WHERE ORG_ID = 'O001'; -- If O001 exists in DEPARTMENT

Result:
ORA-02292: integrity constraint violated - child record found

Similarly, if a DCODE in DEPARTMENT is referenced in EMPLOYEE, deleting it should fail.

SQL Commands:

DELETE FROM DEPARTMENT WHERE DCODE = 'D001'; -- If D001 exists in EMPLOYEE

Result:
ORA-02292: integrity constraint violated - child record found

Problem Statement 3: Create a table having empcode , Name, deptname, & basic From the existing tables along with the records of the employee who are in a particular department (say, d1) and with a basic Rs. 70000/-

b) From the existing table, add the employees with the basic salary greater than or equal to 70000/-

c) Alter the table to add a net pay column.

d) Replace net pay with 1.5* Basic.

e) Try to remove the net net pay column.

SQL Commands:

INSERT INTO ORGANIZATION (ORG_ID) VALUES ('ORG01');

INSERT INTO DEPARTMENT (DCODE, DNAME, ORG_ID) VALUES ('D001', 'HR', 'ORG01');

INSERT INTO DEPARTMENT (DCODE, DNAME, ORG_ID) VALUES ('D002', 'Finance', 'ORG01');

INSERT INTO DEPARTMENT (DCODE, DNAME, ORG_ID) VALUES ('D003', 'IT', 'ORG01');

-- Multiple employees in HR (D001)

INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E001', 'JOHN DOE', '123 Street', 'New York', 70000, 'D001');

INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E002', 'JANE SMITH', '456 Avenue', 'LA', 80000, 'D001');

-- One employee in Finance (D002)

INSERT INTO EMPLOYEE (ECODE, ENAME, ADDRESS, CITY, BASIC, DCODE)

VALUES ('E003', 'ROBERT BROWN', '789 Blvd', 'Chicago', 90000, 'D002');

-- No employees in IT (D003)

-- E001 takes Casual Leave (CL)

INSERT INTO LEAVE_REGISTER(LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES('CL', TO_DATE('01-OCT-2024'), TO_DATE('01-DEC-2024'), 'E001');

```
-- E002 takes Earned Leave (EL)
INSERT INTO LEAVE_REGISTER(LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES('EL', TO_DATE('02-MAY-2024'), '02-AUG-2024', 'E002');

-- E003 takes Medical Leave (ML)
INSERT INTO LEAVE_REGISTER(LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES('ML', '03-JAN-2024', '03-OCT-2024', 'E003');
-- E001 takes Earned Leave (EL) from 01-Feb-2024 to 10-Feb-2024
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES ('EL', TO_DATE('01-FEB-2024', 'DD-MON-YYYY'), TO_DATE('10-FEB-
2024', 'DD-MON-YYYY'), 'E001');

-- E001 takes Medical Leave (ML) from 15-Mar-2024 to 20-Mar-2024
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES ('ML', TO_DATE('15-MAR-2024', 'DD-MON-YYYY'), TO_DATE('20-MAR-
2024', 'DD-MON-YYYY'), 'E001');

-- E002 takes Casual Leave (CL) from 05-Apr-2024 to 07-Apr-2024
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES ('CL', TO_DATE('05-APR-2024', 'DD-MON-YYYY'), TO_DATE('07-APR-
2024', 'DD-MON-YYYY'), 'E002');

-- E002 takes Earned Leave (EL) from 10-Aug-2024 to 12-Aug-2024
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES ('EL', TO_DATE('10-AUG-2024', 'DD-MON-YYYY'), TO_DATE('12-AUG-
2024', 'DD-MON-YYYY'), 'E002');

-- E003 takes Casual Leave (CL) from 15-Nov-2024 to 20-Nov-2024
INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)
VALUES ('CL', TO_DATE('15-NOV-2024', 'DD-MON-YYYY'), TO_DATE('20-NOV-
2024', 'DD-MON-YYYY'), 'E003');
```

```
-- E003 takes Earned Leave (EL) from 10-Feb-2024 to 15-Feb-2024

INSERT INTO LEAVE_REGISTER (LEAVE_TYPE, FROM_DT, TO_DT, ECODE)

VALUES ('EL', TO_DATE('10-FEB-2024', 'DD-MON-YYYY'), TO_DATE('15-FEB-
2024', 'DD-MON-YYYY'), 'E003');

SELECT * FROM ORGANIZATION;

SELECT * FROM EMPLOYEE;

SELECT * FROM DEPARTMENT;

SELECT * FROM LEAVE_REGISTER;


CREATE TABLE TEMP AS

SELECT ECODE,ENAME,DNAME,BASIC FROM EMPLOYEE NATURAL JOIN
DEPARTMENT WHERE DCODE='D001' AND BASIC>70000;


SELECT * FROM TEMP;


INSERT INTO TEMP (ECODE, ENAME, DNAME, BASIC)

SELECT ECODE, ENAME, DNAME, BASIC

FROM EMPLOYEE NATURAL JOIN DEPARTMENT

WHERE BASIC > 70000 AND DCODE<>'D001';



ALTER TABLE TEMP

ADD(NETPAY FLOAT);


UPDATE TEMP SET NETPAY=1.5*BASIC;


SELECT * FROM TEMP;


ALTER TABLE TEMP DROP COLUMN NETPAY;


SELECT * FROM TEMP;
```

Output:

```
ORG_I
-----
ORG01
```

```
ECODE ENAME                ADDRESS                CITY          BASIC
----- -------------------- ---------------------- ---------- ----------
JN_DT     DCODE
--------- -----
E001  JOHN DOE             123 Street             New York       70000
18-MAR-25 D001

E002  JANE SMITH           456 Avenue             LA             80000
18-MAR-25 D001
E003  ROBERT BROWN         789 Blvd               Chicago        90000
18-MAR-25 D002
```

```
DCODE DNAME                     ORG_I
----- ------------------------- -----
D001  HR                        ORG01
D002  Finance                   ORG01
D003  IT                        ORG01
```

```
LEAVE FROM_DT   TO_DT     ECODE
----- --------- --------- -----
CL    01-OCT-24 01-DEC-24 E001
EL    02-MAY-24 02-AUG-24 E002
ML    03-JAN-24 03-OCT-24 E003
EL    01-FEB-24 10-FEB-24 E001
ML    15-MAR-24 20-MAR-24 E001
CL    05-APR-24 07-APR-24 E002
EL    10-AUG-24 12-AUG-24 E002
CL    15-NOV-24 20-NOV-24 E003
```

```
EL    10-FEB-24 15-FEB-24 E003
```

```
Table created.
```

ECODE	ENAME	DNAME	BASIC
E002	JANE SMITH	HR	80000

ECODE	ENAME	DNAME	BASIC	NETPAY
E002	JANE SMITH	HR	80000	120000
E003	ROBERT BROWN	Finance	90000	135000

ECODE	ENAME	DNAME	BASIC
E002	JANE SMITH	HR	80000
E003	ROBERT BROWN	Finance	90000

Problem Statement 4: In a library, for each book book-id, serial number (denotes copy number of a book), title, author, publisher and price are stored. Book-id and serial number together will be unique identifier for a book. Members are either student or faculty. Each member has unique member-id. Name, e-mail, address are also to be stored. Maximum number of books that a member can retain depends on member type. There may be other such parameters that depend on member type. Design should be flexible. For any transaction (book issue or return), members are supposed to place transactions slip. Each Transaction will have a unique id. User will submit member-id, book-id, and serial number (only for book return). Design and create the tables to store the book, member and transaction information. When a book is issued to a member a field like, To_Be_Returned_By has to be set as DT_Issue + 7 days. At the time of book return, DT_Return will store the actual return date. While new book arrives, serial number will be last serial number for the Book-id +1. System should also keep track of the status of each physical book – whether issued or available. Design and create the tables. Populate the database maintaining the logic of transaction. May be issue/return will need update in multiple tables.

Write down the queries for the following:

a) Display total number of copies (irrespective of issued or not) for each book in the library and number of such copies issued

b) Find the members holding the books even after due date

c) Find the transaction details for delayed book returns and delay in terms of number of days.

d) Find the student members not making any transaction and do the same for faculty members.

e) Find the count of issue for each book (not the specific copy).

CODE:

```
-- 1. Create Table: MEMBER (Superclass)
CREATE TABLE MEMBER (
    MEMBER_ID CHAR(5) PRIMARY KEY,
    NAME VARCHAR(50) NOT NULL,
    EMAIL VARCHAR(50) UNIQUE,
    ADDRESS VARCHAR(100)
);

-- 2. Create Table: STUDENT (Subclass of MEMBER)
CREATE TABLE STUDENT (
    MEMBER_ID CHAR(5) PRIMARY KEY,
    MAX_BOOKS INT CHECK (MAX_BOOKS > 0),
    COURSE VARCHAR(50),
    FOREIGN KEY (MEMBER_ID) REFERENCES MEMBER(MEMBER_ID)
```

```
);

-- 3. Create Table: FACULTY (Subclass of MEMBER)
CREATE TABLE FACULTY (
    MEMBER_ID CHAR(5) PRIMARY KEY,
    MAX_BOOKS INT CHECK (MAX_BOOKS > 0),
    DEPARTMENT VARCHAR(50),
    FOREIGN KEY (MEMBER_ID) REFERENCES MEMBER(MEMBER_ID)
);

-- 4. Create Table: BOOK
CREATE TABLE BOOK (
    BOOK_ID CHAR(5) PRIMARY KEY,
    TITLE VARCHAR(100) NOT NULL,
    AUTHOR VARCHAR(50) NOT NULL,
    PUBLISHER VARCHAR(50),
    PRICE DECIMAL(8,2) NOT NULL
);

-- 5. Create Table: BOOK_COPY
CREATE TABLE BOOK_COPY (
    BOOK_ID CHAR(5),
    SERIAL_NO INT,
    STATUS VARCHAR(10) CHECK (STATUS IN ('Available', 'Issued')),
    PRIMARY KEY (BOOK_ID, SERIAL_NO),
    FOREIGN KEY (BOOK_ID) REFERENCES BOOK(BOOK_ID)
);

-- 6. Create Table: TRANSACTION
CREATE TABLE TRANSACTION (
    TRANS_ID CHAR(10) PRIMARY KEY,
    MEMBER_ID CHAR(5) NOT NULL,
    BOOK_ID CHAR(5) NOT NULL,
    SERIAL_NO INT NOT NULL,
    DT_ISSUE DATE DEFAULT CURRENT_DATE,
    TO_BE_RETURNED_BY DATE DEFAULT (CURRENT_DATE + INTERVAL '7'
DAY),
    DT_RETURN DATE NULL,
    FOREIGN KEY (MEMBER_ID) REFERENCES MEMBER(MEMBER_ID),
    FOREIGN KEY (BOOK_ID, SERIAL_NO) REFERENCES
BOOK_COPY(BOOK_ID, SERIAL_NO)
);

-- Insert Sample Data into MEMBER
INSERT INTO MEMBER VALUES ('M001', 'John Doe', 'john@example.com', '123
Main St');
```

```
INSERT INTO MEMBER VALUES ('M002', 'Dr. Smith', 'smith@example.edu', '456
Elm St');
INSERT INTO MEMBER VALUES ('M003', 'Alice Johnson', 'alice@example.com',
'789 Pine St');

-- Insert Sample Data into STUDENT (Subclass of MEMBER)
INSERT INTO STUDENT VALUES ('M001', 3, 'Computer Science');
INSERT INTO STUDENT VALUES ('M003', 3, 'Mathematics');

-- Insert Sample Data into FACULTY (Subclass of MEMBER)
INSERT INTO FACULTY VALUES ('M002', 5, 'Physics Department');

-- Insert Sample Data into BOOK table
INSERT INTO BOOK VALUES ('B001', 'The Great Gatsby', 'F. Scott Fitzgerald',
'Scribner', 399.99);
INSERT INTO BOOK VALUES ('B002', '1984', 'George Orwell', 'Penguin', 499.50);
INSERT INTO BOOK VALUES ('B003', 'To Kill a Mockingbird', 'Harper Lee', 'J.B.
Lippincott', 299.75);

-- Insert Sample Data into BOOK_COPY table
INSERT INTO BOOK_COPY VALUES ('B001', 1, 'Available');
INSERT INTO BOOK_COPY VALUES ('B001', 2, 'Available');
INSERT INTO BOOK_COPY VALUES ('B002', 1, 'Available');
INSERT INTO BOOK_COPY VALUES ('B002', 2, 'Available');
INSERT INTO BOOK_COPY VALUES ('B003', 1, 'Available');

-- Issue a Book (Insert into TRANSACTION and update BOOK_COPY)
INSERT INTO TRANSACTION (TRANS_ID, MEMBER_ID, BOOK_ID, SERIAL_NO,
DT_ISSUE, TO_BE_RETURNED_BY)
VALUES ('T1001', 'M001', 'B001', 1, CURRENT_DATE, CURRENT_DATE +
INTERVAL '7' DAY);

UPDATE BOOK_COPY SET STATUS = 'Issued' WHERE BOOK_ID = 'B001' AND
SERIAL_NO = 1;

-- Another Book Issue
INSERT INTO TRANSACTION (TRANS_ID, MEMBER_ID, BOOK_ID, SERIAL_NO,
DT_ISSUE, TO_BE_RETURNED_BY)
VALUES ('T1002', 'M002', 'B002', 2, CURRENT_DATE, CURRENT_DATE +
INTERVAL '7' DAY);

UPDATE BOOK_COPY SET STATUS = 'Issued' WHERE BOOK_ID = 'B002' AND
SERIAL_NO = 2;

-- Return Book (Update TRANSACTION and BOOK_COPY)
UPDATE TRANSACTION
```

```
SET DT_RETURN = CURRENT_DATE
WHERE TRANS_ID = 'T1001';

UPDATE BOOK_COPY
SET STATUS = 'Available'
WHERE BOOK_ID = 'B001' AND SERIAL_NO = 1;

-- Check all books
SELECT * FROM BOOK;
SELECT * FROM BOOK_COPY;
SELECT * FROM MEMBER;
SELECT * FROM STUDENT;
SELECT * FROM FACULTY;
SELECT * FROM TRANSACTION;
```

OUTPUT:

```
BOOK_
-----
TITLE
--------------------------------------------------------------------------------
AUTHOR
------------------------------------------------
PUBLISHER                                          PRICE
---------------------------------------------- ----------
B001
The Great Gatsby
F. Scott Fitzgerald
Scribner                                           399.99



BOOK_
-----
TITLE
--------------------------------------------------------------------------------
AUTHOR
```

PUBLISHER PRICE

B002

1984

George Orwell

Penguin 499.5

BOOK_

TITLE

AUTHOR

PUBLISHER PRICE

B003

To Kill a Mockingbird

Harper Lee

J.B. Lippincott 299.75

BOOK_ SERIAL_NO STATUS

B001 1 Available

B001 2 Available

B002 1 Available

B002 2 Issued

B003 1 Available

MEMBE NAME

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

EMAIL

ADDRESS

M001 John Doe

john@example.com

123 Main St

M002 Dr. Smith

smith@example.edu

456 Elm St

MEMBE NAME

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

EMAIL

ADDRESS

M003 Alice Johnson

alice@example.com

789 Pine St

MEMBE MAX_BOOKS COURSE

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

M001   3 Computer Science

M003   3 Mathematics

MEMBE MAX_BOOKS DEPARTMENT

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

M002   5 Physics Department

TRANS_ID   MEMBE BOOK_   SERIAL_NO DT_ISSUE TO_BE_RET
DT_RETURN

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

T1001   M001 B001   1 19-MAR-25 26-MAR-25 19-MAR-25

T1002   M002 B002   2 19-MAR-25 26-MAR-25