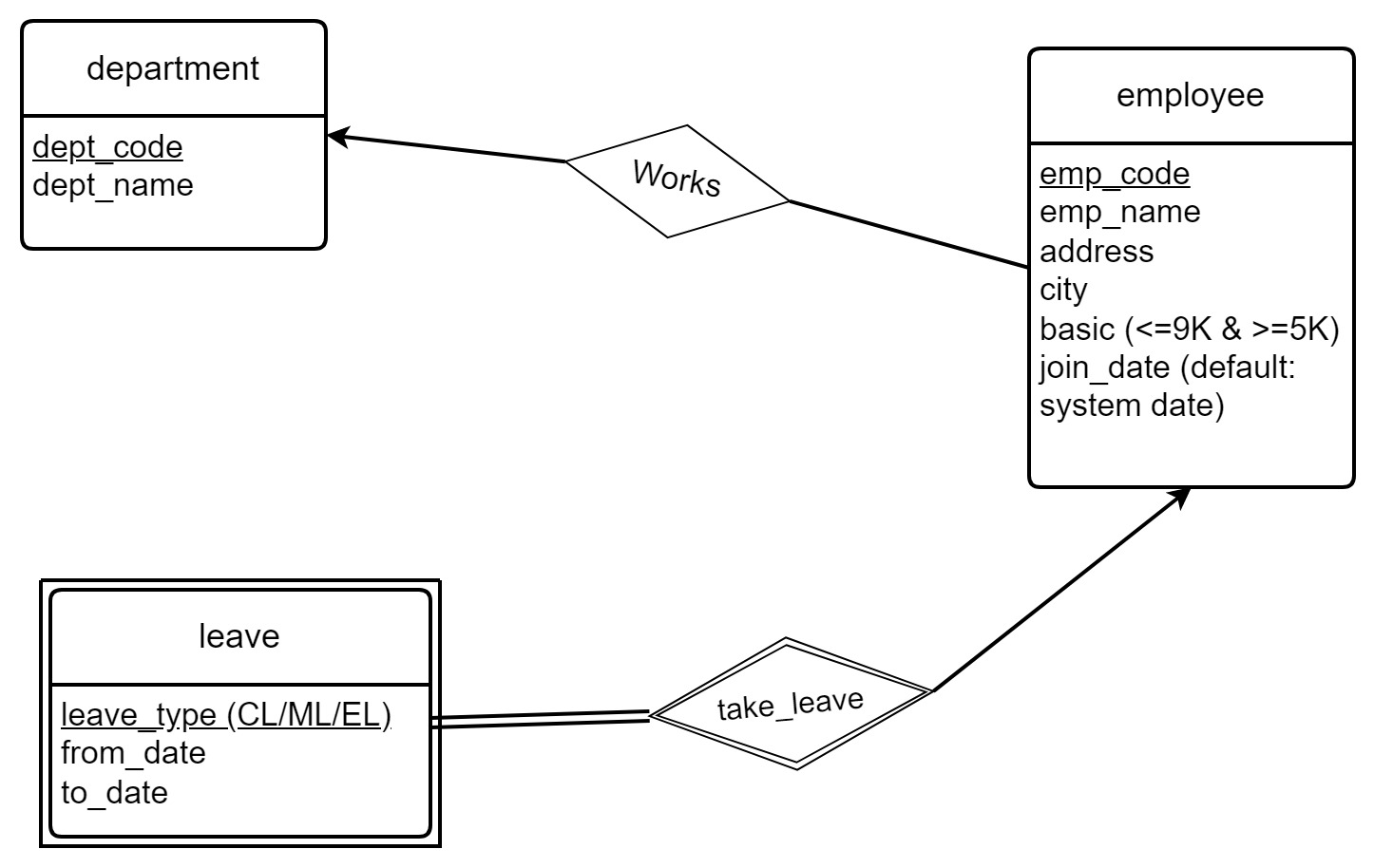
QUESTION NO 1.:

ER Diagram design:

Each **department** has a unique code. Suppose its *dept\_code* and the primary key of department entity. Each department has a name say it is *dept\_name*. Each **employee** has unique code named *emp\_code* and it is primary key of employee entities. For each employee *name, address, basic, city join date* are also stored. As given in the question, each department may have number of employees. We can consider that, one employee may be associated to at most one department. So, we can think of a relationship between employee and department named *Works*. It is a many-to-one relationship from employee to department.

**Leave** information about employees are stored. Attributes for a leave are *leave\_type, from\_date, to\_date*. It is clear that leave is a weak entity set. As leave is dependent on employee who takes the leave. So, there is a weak relationship *take\_leave* between leave and employee. So, employee is an identifying strong entity set for leave and *take\_leave* is identifying relationship. There is no meaning of leave without an employee. So, participation of leave in *take\_leave* is total. The relationship ‘take\_leave’ is many-to-one from leave to employee.

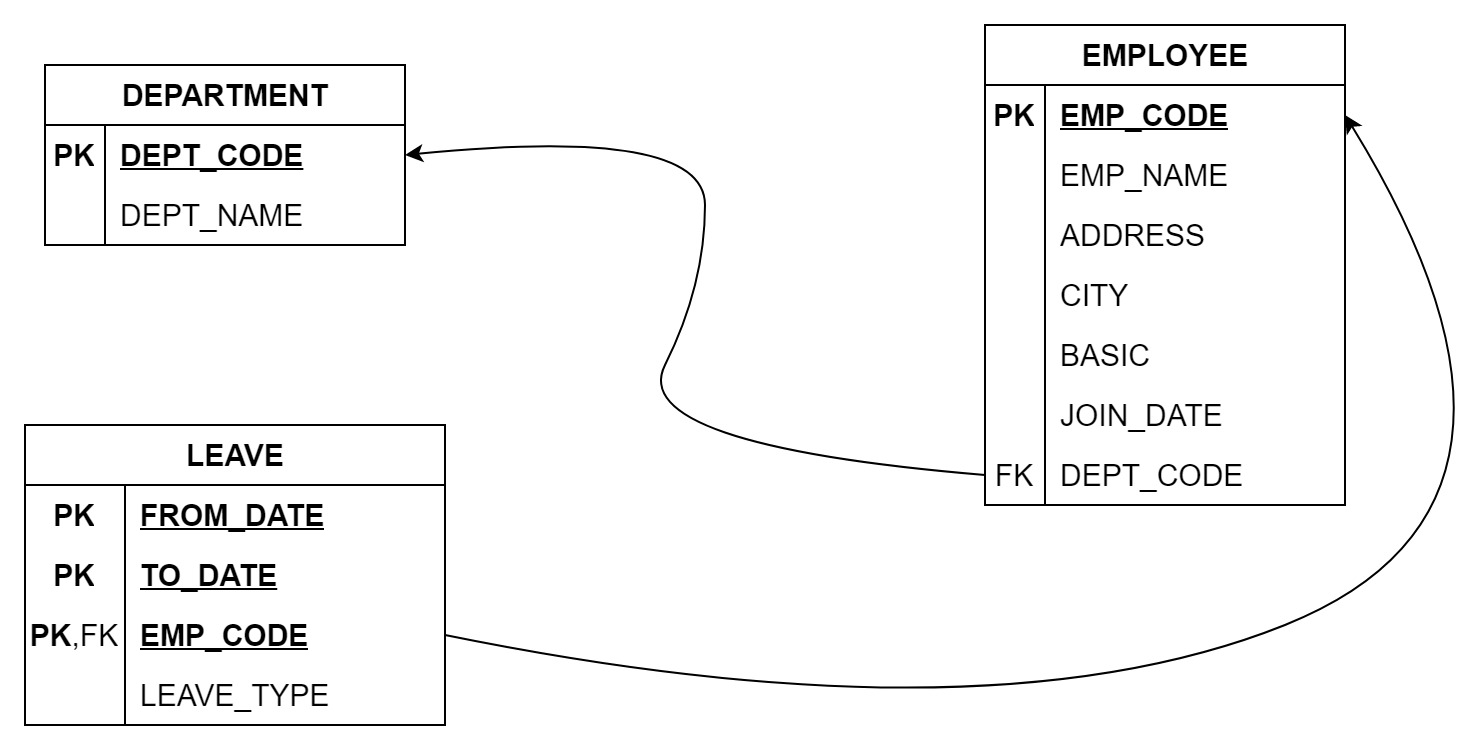
The ER Diagram is given below,



Design and implement Tables(relations):

**A**ccording to the ER diagram depicted above, three tables are created – Employee, Department and Leave. Department has primary key dept\_code and another attribute dept\_name. Each employee has primary key emp\_code and other attributes as emp\_name, basic etc. As each employee can have at most one department as per our design, there will be a foreign key dept\_code referring to the department table. As we know leave is a weak entity set. So, leave table will contain emp\_code from the employee table. Emp\_code in leave table will be foreign key that refers the employee table. And, emp\_code,from\_date,to\_date together will be primary key. Also, after the deletion of an employee, all leave info related to that employee must be deleted(cascaded).

Diagram for table design:



SQL for table design:

CREATE TABLE DEPARTMENT (

DEPT\_CODE CHAR(3) PRIMARY KEY,

DEPT\_NAME CHAR(10) NOT NULL

);

CREATE TABLE EMPLOYEE (

EMP\_CODE CHAR(3) PRIMARY KEY,

EMP\_NAME CHAR(20) NOT NULL,

ADDRESS CHAR(40),

CITY CHAR(10),

BASIC NUMBER(4,0) CHECK (BASIC>=5000 AND BASIC<=9000),

JOIN\_DATE DATE DEFAULT CURRENT\_DATE,

DEPT\_CODE CHAR(3) FOREIGN KEY REFERENCES DEPARTMENT(DEPT\_CODE)

);

CREATE TABLE LEAVE (

LEAVE\_TYPE CHAR(2) CHECK (leave\_type IN ('cl', 'ml', 'el')),

FROM\_DATE DATE,

TO\_DATE DATE,

EMP\_CODE CHAR(3) FOREIGN KEY REFERENCES EMPLOYEE(EMP\_CODE) ON DELETE CASCADE,

PRIMARY KEY(FROM\_DATE, TO\_DATE, EMP\_CODE)

);

QUESTION NO 3.:

A )) 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. 7000/-

CREATE TABLE EMP\_DEPT\_BASIC AS

SELECT e.EMP\_CODE, e.EMP\_NAME, d.DEPT\_NAME, e.BASIC

FROM EMPLOYEE AS e

INNER JOIN DEPARTMENT AS d ON e.DEPT\_CODE = d.DEPT\_CODE

WHERE d.DEPT\_CODE = 'd1' AND e.BASIC = 7000;

B )) From the existing table, add the employees with the basic salary greater than or equal to 7000/-

INSERT INTO EMP\_DEPT\_BASIC (EMP\_CODE, EMP\_NAME, DEPT\_NAME, BASIC)

SELECT e.EMP\_CODE, e.EMP\_NAME, d.DEPT\_NAME, e.BASIC

FROM EMPLOYEE AS e

INNER JOIN DEPARTMENT AS d ON e.DEPT\_CODE = d.DEPT\_CODE

WHERE e.BASIC >= 7000;

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

ALTER TABLE EMP\_DEPT\_BASIC

ADD NET\_PAY NUMBER(8,2);

D )) Replace net pay with 1.5\* Basic.

UPDATE EMP\_DEPT\_BASIC

SET NET\_PAY = 1.5 \* BASIC;

E )) Try to remove the net net pay column.

ALTER TABLE EMP\_DEPT\_BASIC

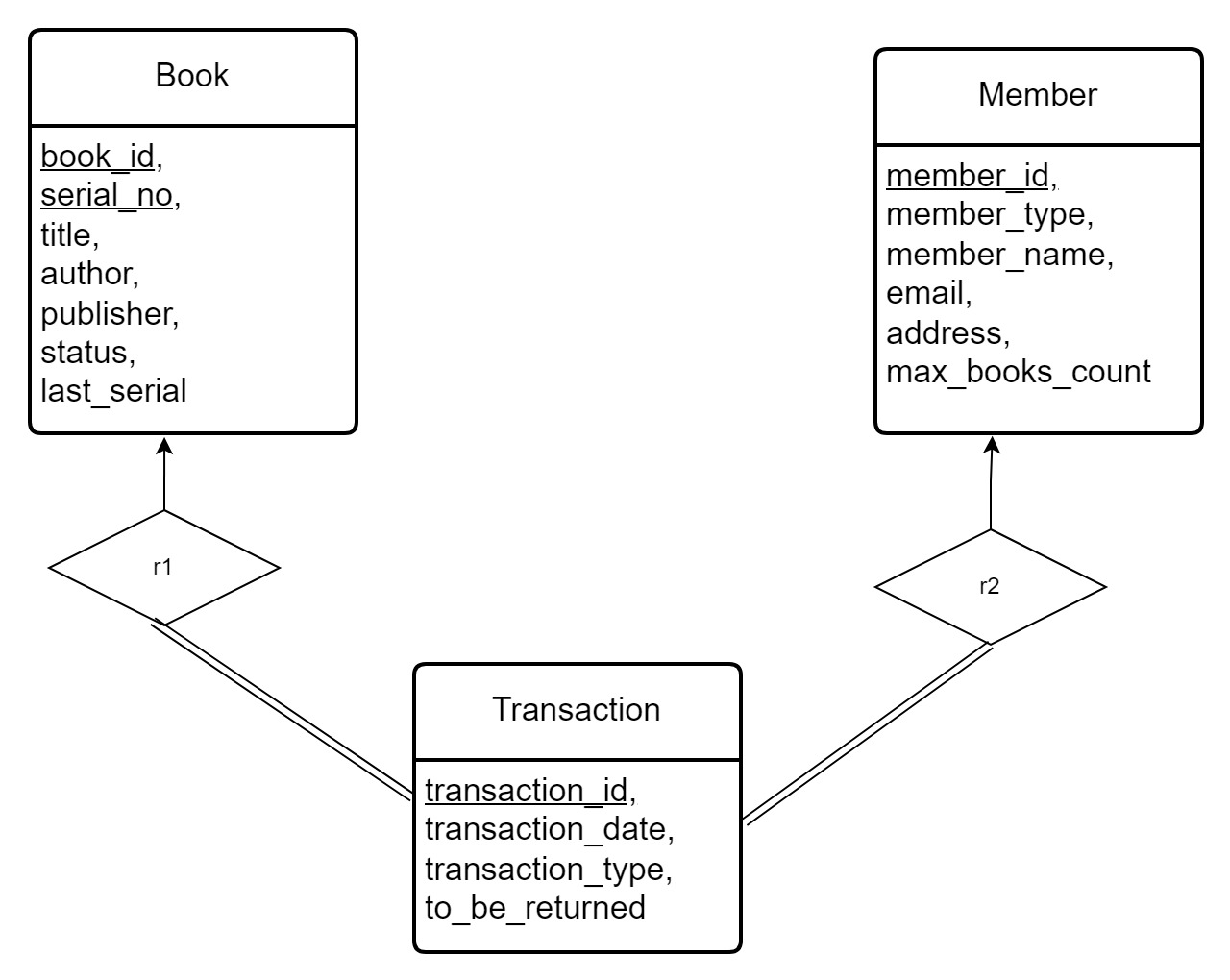
DROP COLUMN NET\_PAY;

QUESTION NO 5.:

ER Diagram design:

Each Book has book-id, serial-no both taken as unique primary key. Also other info related to book as title, author, publisher, status, last\_serial etc are there. Each Member is uniquely identified by primary key member-id. Each member also have attributes such as name, member-type, email, address, max-no-books etc. Each Transaction has unique transaction-id which is the primary key. Each transaction entity also holds transaction\_date(dt\_issue or dt\_return), transaction\_type(issue or return), to\_be\_returned(to be used for during return). There is a many(total) to one relationship from Transaction to Book. Similarly from Transaction to member.

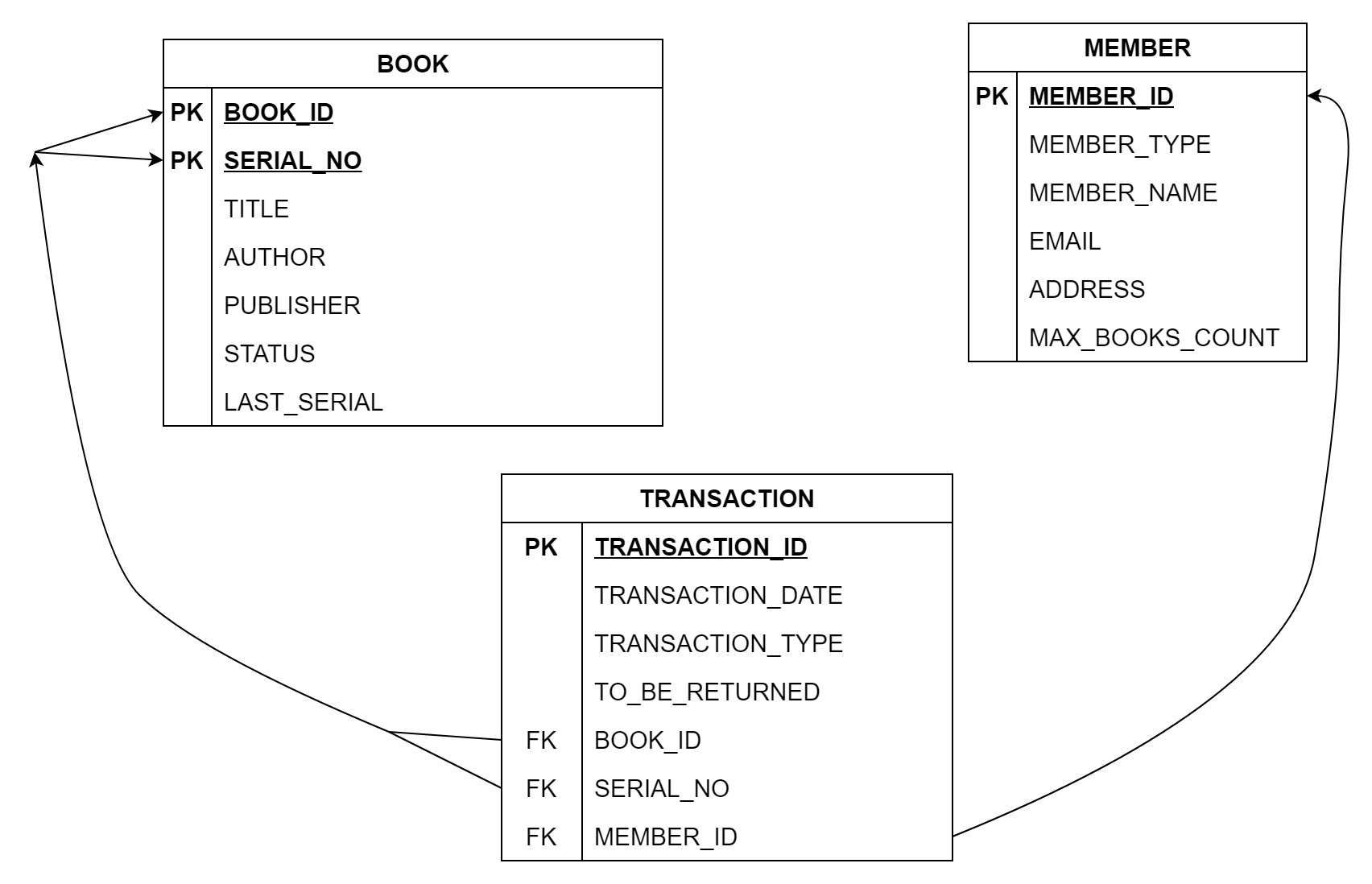
The ER Diagram is given below,



Design and implement Tables(relations):

**A**ccording to the ER diagram shown above, we create three tables – Book, Transaction and Member. Book has primary key as book\_id and serial\_no together, and some other attributes are there. Transaction and Member have primary key as transaction\_id and member\_id respectively. Other attributes are also present. Also, we export book\_id, serial\_no into Transaction table, and they work as foreign key to refer the Book table. Similarly, we include member\_id in the Transaction table. This is used as foreign key to refer the Member table.

The diagram for table design is given –



SQL for table design:

CREATE TABLE BOOK (

BOOK\_ID CHAR(3),

SERIAL\_NO NUMBER(3,0),

TITLE CHAR(100),

AUTHOR CHAR(100),

PUBLISHER CHAR(100),

STATUS CHAR(10) CHECK (STATUS IN (‘issued’,’available’))

LAST\_SERIAL NUMBER(3,0),

PRIMARY KEY(BOOK\_ID, SERIAL\_NO)

);

CREATE TABLE MEMBER (

MEMBER\_ID CHAR(3) PRIMARY KEY,

MEMBER\_TYPE CHAR(7) CHECK (MEMBER\_TYPE IN (‘student’,’faculty’)),

MEMBER\_NAME CHAR(20) NOT NULL,

EMAIL CHAR(50),

ADDRESS CHAR(100),

MAX\_BOOKS\_COUNT NUMBER(1,0)

);

CREATE TABLE TRANSACTION (

TRANSACTION\_ID CHAR(3) PRIMARY KEY,

TRANSACTION\_DATE DATE,

TRANSACTION\_TYPE CHAR(6) CHECK (TRANSACTION\_TYPE IN (‘issue’,’return’)),

TO\_BE\_RETURNED DATE DATE,

BOOK\_ID CHAR(3),

SERIAL\_NO NUMBER(3,0),

MEMBER\_ID CHAR(3),

FOREIGN KEY(BOOK\_ID, SERIAL\_NO) REFERENCES BOOK(BOOK\_ID,SERIAL\_NO),

FOREIGN KEY(MEMBER\_ID) REFERENCES MEMBER(MEMBER\_ID)

);

Related Questions:

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

SELECT B.BOOK\_ID, B.TITLE, COUNT(\*) AS TOTAL\_COPIES, COUNT(T.BOOK\_ID) AS ISSUED\_COPIES

FROM BOOK B

LEFT JOIN TRANSACTION T

ON B.BOOK\_ID = T.BOOK\_ID AND B.SERIAL\_NO = T.SERIAL\_NO

GROUP BY B.BOOK\_ID, B.TITLE;

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

SELECT m.member\_id, m.member\_name, t.book\_id, t.serial\_no

FROM MEMBER AS m

INNER JOIN TRANSACTION AS t

ON m.member\_id = t.member\_id

WHERE t.transaction\_type = ‘issue’

AND t.to\_be\_returned < CURRENT\_DATE;

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

SELECT T.TRANSACTION\_ID, T.TRANSACTION\_DATE, T.BOOK\_ID, T.SERIAL\_NO, T.MEMBER\_ID, T.TO\_BE\_RETURNED,

B.TITLE, DATEDIFF(DAY, T.TO\_BE\_RETURNED, GETDATE()) AS DAYS\_DELAYED

FROM TRANSACTION AS T

INNER JOIN BOOK AS B

ON T.BOOK\_ID = B.BOOK\_ID AND T.SERIAL\_NO = B.SERIAL\_NO

WHERE T.TRANSACTION\_TYPE = 'return' AND T.TO\_BE\_RETURNED < GETDATE();

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

SELECT \* FROM MEMBER

LEFT JOIN TRANSACTION

ON MEMBER.MEMBER\_ID = TRANSACTION.MEMBER\_ID

WHERE TRANSACTION.TRANSACTION\_ID IS NULL

AND MEMBER.MEMBER\_TYPE = 'student';

SELECT \* FROM MEMBER

LEFT JOIN TRANSACTION

ON MEMBER.MEMBER\_ID = TRANSACTION.MEMBER\_ID

WHERE TRANSACTION.TRANSACTION\_ID IS NULL

AND MEMBER.MEMBER\_TYPE =’faculty’;

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

SELECT BOOK\_ID, COUNT(\*) AS ISSUE\_COUNT

FROM TRANSACTION

WHERE TRANSACTION\_TYPE = 'issue'

GROUP BY BOOK\_ID;