TIS1101 Database Fundamentals

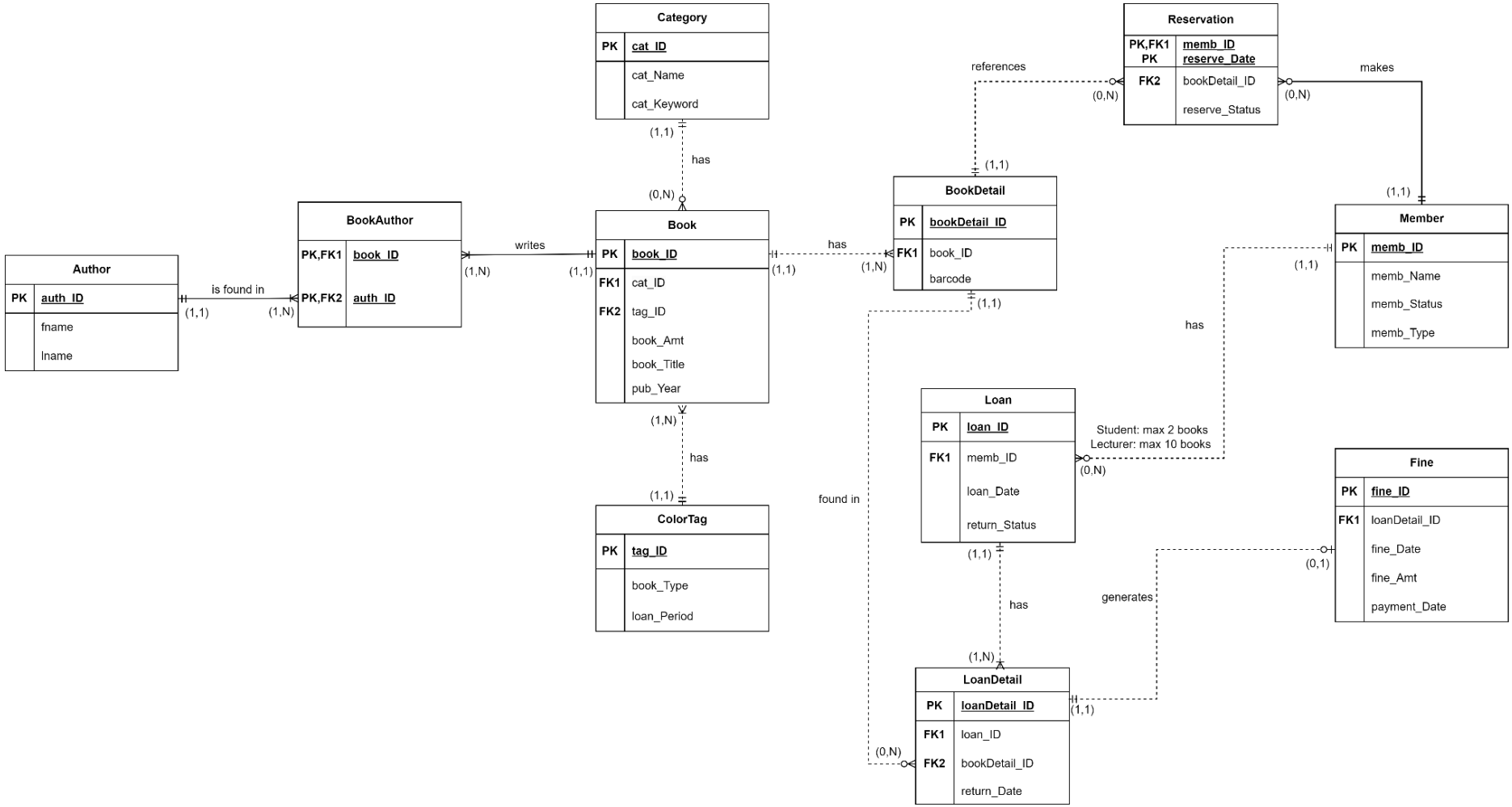
Assignment 2

Title: 5 - University Library System

Prepared by:

	Student ID	Name	Email
Leader	1211100925	Ang Jin Nan	1211100925@student.mmu.edu.my
Member	1211102630 1211103311 1211102777	Chan Kar Kin Ng Yun Shi Tai Qi Tong	1211102630@student.mmu.edu.my 1211103311@student.mmu.edu.my 1211102777@student.mmu.edu.my

1. Corrected and normalized ERD



2. Data Dictionary

Table Name	Attribute Name	Contents	Type	Required	PK/FK	FK reference table
Author	auth_ID	author id	varchar(5)	Y	PK	
	fname	author first name	varchar(20)	Y		
	lname	author last name	varchar(20)	Y		
Member	memb_ID	member id	varchar(5)	Y	PK	
	memb_Name	member name	varchar(20)	Y		
	memb_Status	member active status (Active, Inactive, Suspended)	varchar(10)	Y		
	memb_Type	member type (Student, Lecturer)	varchar(10)	Y		
Category	cat_ID	book category id	varchar(5)	Y	PK	
	cat_Name	book category name	varchar(50)	Y		
	cat_Keyword	book category keyword for searching	varchar(20)	Y		
ColorTag	tag_ID	book color tag id	varchar(2)	Y	PK	
	book_Type	book type	varchar(20)	Y		
	loan_Period	book loan period	int	Y		
Book	book_ID	book id	varchar(5)	Y	PK	
	cat_ID	book category id	varchar(5)	Y	FK1	Category
	tag_ID	book color tag id	varchar(2)	Y	FK2	ColorTag
	book_Amt	book copies	smallint	Y		
	book_Title	book title	varchar(50)	Y		
	pub_Year	book publication year	int	Y		

BookAuthor	book_ID	book id	varchar(5)	Y	PK, FK1	Book
	auth_ID	author id	varchar(5)	Y	PK, FK2	Author
BookDetail	bookDetail_ID	book detail id for each book copy	int	Y	PK	
	book_ID	book id	varchar(5)	Y	FK1	Book
	barcode	book barcode	varchar(10)	Y		
Loan	loan_ID	loan id	varchar(5)	Y	PK	
	memb_ID	member id	varchar(5)	Y	FK1	Member
	loan_Date	loan date	date	Y		
	return_Status	book return status (Yes, No)	varchar(10)	Y		
LoanDetail	loanDetail_ID	loan detail id of each book borrowed in a single loan	varchar(5)	Y	PK	
	loan_ID	loan id	varchar(5)	Y	FK1	Loan
	bookDetail_ID	book detail id for each book copy	int	Y	FK2	BookDetail
	return_Date	date of book returned	date			
Reservation	memb_ID	member id	varchar(5)	Y	PK, FK1	Member
	reserve_Date	date of reservation made	date	Y	PK	
	bookDetail_ID	book detail for each book copy	int	Y	FK2	BookDetail
	reserve_Status	book reservation status (Cancelled, Pending, Expired, Completed)	varchar(10)	Y		

Fine	fine_ID	fine id	varchar(5)	Y	PK	
	loanDetail_ID	loan detail id of each book borrowed in a single loan	varchar(5)	Y	FK1	LoanDetail
	fine_Date	date of fine issued	date	Y		
	fine_Amt	fine amount issued	decimal(5,2)	Y		
	payment_Date	payment date of fine	date			

3. Creation of Tables

CREATE TABLE **Author**(
    auth_ID varchar(5) **primary key** not null,
    fname varchar(20) not null,
    lname varchar(20) not null
)

```
db2 => create table author(auth_id varchar(5) primary key not null, fname varchar(20) not null, lname varchar(20) not null)
DB20000I  The SQL command completed successfully.
db2 => select * from author

AUTH_ID FNAME                LNAME
------- -------------------- --------------------

  0 record(s) selected.
```

CREATE TABLE **Member**(
    memb_ID varchar(5) **primary key** not null,
    memb_Name varchar(20) not null,
    memb_Status varchar(10) not null,
    memb_Type varchar(10) not null,
    CONSTRAINT check_memb_status **CHECK**(memb_status in ('Inactive', 'Active', 'Suspended')),
    CONSTRAINT check_memb_type **CHECK**(memb_type in ('Student', 'Lecturer'))
)

```
db2 => create table member(memb_id varchar(5) primary key not null, memb_name varchar(20) not null, memb_status varchar(10) not null,
 memb_type varchar(10) not null, constraint check_memb_status check (memb_status in ('Inactive', 'Active', 'Suspended')), constraint
check_memb_type check(memb_type in ('Student', 'Lecturer')))
DB20000I  The SQL command completed successfully.
db2 => select * from member

MEMB_ID MEMB_NAME            MEMB_STATUS MEMB_TYPE
------- -------------------- ----------- ----------

  0 record(s) selected.
```

CREATE TABLE **ColorTag**(
    tag_ID varchar(2) **primary key** not null,
    book_Type varchar(20) not null,
    loan_Period int not null
)

```
db2 => create table colortag(tag_id varchar(2) primary key not null, book_type varchar(20) not n
ull, loan_period int not null)
DB20000I  The SQL command completed successfully.
db2 => select * from colortag

TAG_ID BOOK_TYPE            LOAN_PERIOD
------ -------------------- -----------

  0 record(s) selected.
```

CREATE TABLE **Category**(
    cat_ID varchar(5) **primary key** not null,
    cat_Name varchar(50) not null,
    cat_Keyword varchar(20) not null
)

```
db2 => create table category(cat_id varchar(5) primary key not null, cat_name varchar(50) not nu
ll, cat_keyword varchar(20) not null)
DB20000I  The SQL command completed successfully.
db2 => select * from category

CAT_ID CAT_NAME                                           CAT_KEYWORD
------ -------------------------------------------------- --------------------

  0 record(s) selected.
```

CREATE TABLE **Book**(

book_ID varchar(5) **primary key** not null,

cat_ID varchar(5) not null,

tag_ID varchar(2) not null,

book_Amt **smallint** not null,

book_Title varchar(50) not null,

pub_Year int not null,

FOREIGN KEY(cat_ID) REFERENCES Category,

FOREIGN KEY(tag_ID) REFERENCES ColorTag

)

```
db2 => create table book(book_id varchar(5) primary key not null, cat_id varchar(5) not null, t
ag_id varchar(2) not null, book_amt smallint not null, book_title varchar(50) not null, pub_yea
r int not null, foreign key(cat_id) references category, foreign key(tag_id)references colortag
)
DB20000I  The SQL command completed successfully.
db2 =>
db2 => select * from book

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                         PUB_YEAR
------- ------ ------ -------- -------------------------------------------------- -----------

  0 record(s) selected.
```

CREATE TABLE **BookAuthor**(

book_ID varchar(5) not null,

auth_ID varchar(5) not null,

primary key(book_ID, auth_ID),

FOREIGN KEY(book_ID) REFERENCES Book,

FOREIGN KEY(auth_ID) REFERENCES Author

)

```
db2 => Create table bookAuthor(book_id varchar(5) not null, auth_id varchar(5) not null
, primary key(book_id, auth_id), foreign key(book_id) references book, foreign key(auth
_id) references author)
DB20000I  The SQL command completed successfully.
db2 => select * from bookauthor

BOOK_ID AUTH_ID
------- -------

  0 record(s) selected.
```

CREATE TABLE **BookDetail**(

bookDetail_ID int generated always as identity(start with 1, increment by 1) **primary key** not null,

book_ID varchar(5) not null,

barcode varchar(10) UNIQUE not null,

FOREIGN KEY(book_ID) REFERENCES Book

)

```
db2 => create table bookDetail(bookdetail_id int generated always as identity(start
 with 1, increment by 1) primary key not null, book_id varchar(5) not null, barcode
 varchar(10) unique not null, foreign key(book_id)references book)
DB20000I  The SQL command completed successfully.
db2 => SELECT * FROM BOOKDETAIL

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------

  0 record(s) selected.
```

CREATE TABLE **Loan**(

loan_ID varchar(5) **primary key** not null,

memb_ID varchar(5) not null,

loan_Date date **DEFAULT** current date,

return_Status varchar(10) not null **DEFAULT** ‘No’ **CHECK**(return_Status in ('Yes','No')),

FOREIGN KEY (memb_ID) REFERENCES Member

)

```
db2 => create table loan(loan_id varchar(5) primary key not null, memb_id varchar(5) not n
ull, loan_date date default current date, return_status varchar(10) not null default 'No'
check(return_status in ('Yes','No')), foreign key (memb_id) references member)
DB20000I  The SQL command completed successfully.
db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------

  0 record(s) selected.
```

CREATE TABLE **LoanDetail**(
loanDetail_ID varchar(5) **primary key** not null,
loan_ID varchar(5) not null,
bookDetail_ID int not null,
return_Date date **DEFAULT** null,
FOREIGN KEY(loan_ID) REFERENCES Loan,
FOREIGN KEY(bookDetail_ID) REFERENCES BookDetail
)

```
db2 => create table loandetail(loandetail_id varchar(5) primary key not null, loan_id varchar
(5) not null, bookdetail_id int not null, return_date date default null, foreign key(loan_id)
references loan, foreign key(bookdetail_id)references bookdetail)
DB20000I  The SQL command completed successfully.
db2 => select * From loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------

  0 record(s) selected.
```

CREATE TABLE **Reservation**(
memb_ID varchar(5) not null,
reserve_Date date not null,
bookDetail_ID int not null,
reserve_Status varchar(10) **DEFAULT** 'Pending' **CHECK**(reserve_status in ('Cancelled', 'Pending', 'Expired', 'Completed')),
primary key(memb_ID,reserve_Date),
FOREIGN KEY (memb_ID) REFERENCES Member,
FOREIGN KEY (bookDetail_ID) REFERENCES BookDetail
)

```
db2 => create table reservation(memb_id varchar(5) not null,reserve_date date not null, bookd
etail_id int not null, reserve_status varchar(10) default 'Pending' check(reserve_status in (
'Cancelled','Pending','Expired','Completed')), primary key(memb_id,reserve_date), foreign key
 (memb_id) references member, foreign key(bookdetail_id) references bookDetail)
DB20000I  The SQL command completed successfully.
db2 => select  * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------

  0 record(s) selected.
```

CREATE TABLE **Fine**(
fine_ID varchar(5) **primary key** not null,
loanDetail_ID varchar(5) not null UNIQUE,
fine_Date date **DEFAULT** current date not null,
fine_Amt decimal(5, 2) not null,
payment_Date date **DEFAULT** null,
FOREIGN KEY (loanDetail_ID) REFERENCES LoanDetail
)

```
db2 => CREATE TABLE fine (fine_id VARCHAR(5) primary key NOT NULL, loandetail_id VARCHAR(5) N
OT NULL unique, fine_date DATE DEFAULT CURRENT DATE NOT NULL, fine_amt DECIMAL(5, 2) NOT NULL
, payment_date DATE DEFAULT NULL, FOREIGN KEY (loandetail_id) REFERENCES loandetail)
DB20000I  The SQL command completed successfully.
db2 => select * From fine

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------

  0 record(s) selected.
```

```
db2 => list tables

Table/View                      Schema          Type
------------------------------- --------------- -----
AUTHOR                          DB2ADMIN        T
BOOK                            DB2ADMIN        T
BOOKAUTHOR                      DB2ADMIN        T
BOOKDETAIL                      DB2ADMIN        T
CATEGORY                        DB2ADMIN        T
COLORTAG                        DB2ADMIN        T
FINE                            DB2ADMIN        T
LOAN                            DB2ADMIN        T
LOANDETAIL                      DB2ADMIN        T
MEMBER                          DB2ADMIN        T
RESERVATION                     DB2ADMIN        T

  11 record(s) selected.
```

4. Data Insertion

Author

```
db2 => select * From author

AUTH_ID FNAME                LNAME
------- -------------------- --------------------
A001    Winson               Lim
A002    Jennifer             Chan
A003    Jason                Ang
A004    Jacky                Ng
A005    Sophia               Tai
A006    Peter                Su
A007    Wai Chin             Chan
A008    Rachel               Aaron
A009    Edward               Abbey
A010    Megan                Abbott

  10 record(s) selected.
```

Member

```
db2 => select * From member

MEMB_ID MEMB_NAME            MEMB_STATUS MEMB_TYPE
------- -------------------- ----------- ----------
M001    Yun Shi              Active      Student
M002    Qi Tong              Active      Student
M003    Jin Nan              Active      Student
M004    Kar Kin              Inactive    Student
M005    Joshua               Inactive    Lecturer
M006    Lily                 Active      Lecturer
M007    Kai Sheng            Inactive    Lecturer

  7 record(s) selected.
```

Category

```
db2 => select  * from category

CAT_ID CAT_NAME                                           CAT_KEYWORD
------ -------------------------------------------------- --------------------
C001   Information Technology                             IT
C002   Computer Science                                   CS
C003   Law                                                LAW
C004   Mathematics                                        MT
C005   Fiction                                            FIC

  5 record(s) selected.
```

ColorTag

```
db2 => select * from colortag

TAG_ID BOOK_TYPE            LOAN_PERIOD
------ -------------------- -----------
G1     Past Year Paper                1
G2     Research paper                 1
G3     Magazine                       1
B1     Textbook                       5
Y1     Journal                        7
Y2     Others                         7
R1     Reference book                14

  7 record(s) selected.
```

Book

```
db2 => select * from book

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
------- ------ ------ -------- ------------------------------------------------- -----------
CS1     C002   G1            3 Programming Fundamentals                                  2019
CS2     C002   G2            2 The Future of Computer-Assisted Education                 2015
CS3     C002   R1            1 Computer Architecture                                     2019
IT1     C001   B1            2 Textbook of Information Technology                        1988
LAW1    C003   Y1            3 Cambridge Law Journal                                     2021
MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                     2021
CS4     C002   G1            1 Programming Fundamentals                                  2021

  7 record(s) selected.
```

BookAuthor

```
db2 => select  * from bookauthor

BOOK_ID AUTH_ID
------- -------
CS1     A001
CS1     A002
CS1     A008
CS2     A002
CS2     A004
CS3     A001
CS3     A002
CS4     A002
IT1     A002
IT1     A006
LAW1    A003
LAW1    A007
LAW1    A008
LAW1    A009
MT1     A001
MT1     A003
MT1     A008

  17 record(s) selected.
```

BookDetail

```
db2 => select * from bookdetail

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567

  15 record(s) selected.
```

Loan

```
db2 => select  * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 No

  8 record(s) selected.
```

LoanDetail

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 -

  11 record(s) selected.
```

Fine

```
db2 => SELECT * fROM FINE

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------
F1000   LD003         05/22/2023     1.00 05/22/2023
F1001   LD008         06/14/2023     2.00 06/15/2023

  2 record(s) selected.
```

Reservation

```
db2 => select * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------
M001    04/23/2023              2 Expired
M006    04/23/2023              9 Completed
M006    04/30/2023              6 Completed
M003    05/07/2023             10 Expired
M005    05/23/2023             11 Expired
M006    06/01/2023              1 Completed
M001    06/13/2023              4 Expired
M001    06/17/2023              4 Pending

  8 record(s) selected.
```

5. Data Manipulation with SQL:

i. At least one aggregate function (*count*, *max*, *min*, *avg*, *sum*)

(a) Find total number of books in a category

```
SELECT c.cat_name, COUNT(DISTINCT b.book_title) AS distinct_books
FROM category c
JOIN book b ON c.cat_id = b.cat_id
GROUP BY c.cat_name
```

The purpose of this query is to find the total number of books with different book titles in each category. This is done by joining the category and book table based on the cat_id, and then counting the rows. We then group the results by category name.

```
db2 => SELECT c.cat_name, COUNT(distinct b.book_title) AS distinct_books FROM category c JOIN book b ON c.c
at_id = b.cat_id GROUP BY c.cat_name

CAT_NAME                                           DISTINCT_BOOKS
-------------------------------------------------- --------------
Computer Science                                                3
Information Technology                                          1
Law                                                             1
Mathematics                                                     1

  4 record(s) selected.
```

(b) Find latest publication year and earliest publication year of books

```
SELECT DISTINCT book_title,
       MAX(pub_year) AS latest_pub_year,
       MIN(pub_year) AS earliest_pub_year
FROM book
GROUP BY book_title
```

The purpose of this query is to find the latest and earliest publication year of each book from the book table. The MAX() aggregate function is to calculate the maximum value of the pub_year column for the records having the same book_title as latest_pub_year, while MIN() is used to calculate the minimum value of pub_year as earliest_pub_year. The retrieved records are later grouped according to book_title and only distinct book_titles will be displayed.

```
db2 => SELECT DISTINCT book_title, MAX(pub_year) AS latest_pub_year, MIN(pub_year) AS earliest
_pub_year FROM book GROUP BY book_title

BOOK_TITLE                                         LATEST_PUB_YEAR EARLIEST_PUB_YEAR
-------------------------------------------------- --------------- ----------------
Cambridge Law Journal                                         2021             2021
Computer Architecture                                         2019             2019
Mathematic II - Trimester 1 2021/2022                         2021             2021
Programming Fundamentals                                      2021             2019
Textbook of Information Technology                            1988             1988
The Future of Computer-Assisted Education                     2015             2015

  6 record(s) selected.
```

(c) Find the total number of members according to Student, Lecturer, Total Active, Total Suspended and Total Inactive

```
SELECT
    memb_type,
    COUNT(*) AS total_member,
    SUM(CASE WHEN memb_Status = 'Active' THEN 1 ELSE 0 END) AS
Total_Active_Member,
    SUM(CASE WHEN memb_Status = 'Suspended' THEN 1 ELSE 0 END) AS
Total_Suspended_Member,
    SUM(CASE WHEN memb_Status = 'Inactive' THEN 1 ELSE 0 END) AS
Total_Inactive_Member
FROM
    member
WHERE
    memb_type IN ('Student', 'Lecturer')
GROUP BY
    memb_type
```

The purpose of the query is to find the total number of members according to Student, Lecturer, Total Active, Total Suspended and Total Inactive. The COUNT() aggregate function is used to calculate the total members where they member types are in 'Student' and 'Lecturer' while the SUM() aggregate function is used to calculate the number of members with member status 'Active', 'Inactive', and 'Suspended' for each member type with a conditional statement. The results then grouped according to the member type.

```
db2 => select * from member

MEMB_ID MEMB_NAME           MEMB_STATUS MEMB_TYPE
------- ------------------- ----------- ----------
M001    Yun Shi             Active      Student
M002    Qi Tong             Active      Student
M003    Jin Nan             Active      Student
M004    Kar Kin             Inactive    Student
M005    Joshua              Inactive    Lecturer
M006    Lily                Active      Lecturer
M007    Kai Sheng           Inactive    Lecturer

  7 record(s) selected.
```

```
db2 => select memb_type, count(*) as total_member, sum(case when memb_Status = 'Active' then 1 else 0 end) as Total_Active_Member,
 sum(case when memb_Status = 'Suspended' then 1 else 0 end) as Total_Suspended_Member, sum(case when memb_Status = 'Inactive' then
 1 else 0 end) as Total_Inactive_Member from member where memb_type in ('Student', 'Lecturer') group by memb_type

MEMB_TYPE  TOTAL_MEMBER TOTAL_ACTIVE_MEMBER TOTAL_SUSPENDED_MEMBER TOTAL_INACTIVE_MEMBER
---------- ------------ ------------------- ---------------------- --------------------
Lecturer              3                   1                      0                    2
Student               4                   3                      0                    1

  2 record(s) selected.
```

ii. At least one query with a *group by* and *having* clauses

(a) Calculate total number of books loaned and the number of books that have not returned in each month

```
SELECT
    c.cat_Name,
    CAST(MONTHNAME(l.loan_date) AS VARCHAR(20)) AS loan_month,
    COUNT(*) AS total_loan,
    SUM(CASE WHEN ld.return_date IS NULL THEN 1 ELSE 0 END) AS
Books_Not_Returned
FROM
    loan l
JOIN
    loandetail ld ON l.loan_id = ld.loan_id
JOIN
    bookdetail bd ON ld.bookdetail_id = bd.bookdetail_id
JOIN
    book b ON bd.book_id = b.book_id
JOIN
    category c ON b.cat_id = c.cat_id
GROUP BY
    c.cat_Name, CAST(MONTHNAME(l.loan_date) AS VARCHAR(20)),
MONTH(l.loan_date)
HAVING
    COUNT(*) > 0
ORDER BY
    MONTH(l.loan_date)
```

The purpose of the query is to count the total number of books that were loaned and have not returned in each month. The CAST() function is used to convert the data type to varchar(20), which allows the loan_month to be represented as a string. The MONTHNAME() function is then used to retrieve the name of the month from the loan_date value. The count of Books_Not_Returned is calculated by using the SUM() function with a conditional statement. When the return_Date is null, it will consider that the book has not been returned yet. The records are grouped by cat_Name, loan_Month, Total_Loan and Books_Not_Returned, and are ordered by loan_date.

Loan table:

```
db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 Yes
L009    M002    06/15/2023 Yes

  9 record(s) selected.
```

Loan Detail table:

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 06/19/2023
LD012         L009                1 06/19/2023

  12 record(s) selected.
```

Book Detail table:

```
db2 => select * from bookdetail

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567
           16 CS5     82379940

  16 record(s) selected.
```

Result:

```
db2 => SELECT c.cat_Name, CAST(MONTHNAME(l.loan_date) AS VARCHAR(20)) AS loan_month, COUNT(*) AS total_loan, SUM(CASE WHEN ld.return_date IS
NULL THEN 1 ELSE 0 END) AS Books_Not_Returned FROM loan l JOIN loandetail ld ON l.loan_id = ld.loan_id JOIN bookdetail bd ON ld.bookdetail_id
 = bd.bookdetail_id JOIN book b ON bd.book_id = b.book_id JOIN category c ON b.cat_id = c.cat_id GROUP BY c.cat_Name, CAST(MONTHNAME(l.loan_d
ate) AS VARCHAR(20)), MONTH(l.loan_date) HAVING COUNT(*) > 0 ORDER BY MONTH(l.loan_date)

CAT_NAME                                           LOAN_MONTH           TOTAL_LOAN  BOOKS_NOT_RETURNED
-------------------------------------------------- -------------------- ----------- ------------------
Law                                                April                          1                  0
Computer Science                                   May                            2                  0
Computer Science                                   June                           7                  1
Information Technology                             June                           2                  0

  4 record(s) selected.
```

(b) To view top 3 books rented out within a month

```
CREATE VIEW top_3_loan AS
SELECT subq.book_id, subq.book_title, subq.rental_count
FROM (
  SELECT BD.book_id, B.book_title, COUNT(*) AS rental_count
  FROM loan L
  JOIN loandetail LD ON L.loan_id = LD.loan_id
  JOIN bookDetail BD ON LD.bookdetail_id = BD.bookdetail_id
  JOIN book B ON BD.book_id = B.book_id
  WHERE L.loan_date >= TRUNCATE(CURRENT DATE, 'MONTH')
  GROUP BY BD.book_id, B.book_title
  ORDER BY rental_count DESC
  FETCH FIRST 3 ROWS ONLY
) AS subq
```

This view is to list out book_id, book_title and rental count from bookDetail, loan and loandetail tables. It will only show the top 3 books rented out based on the rental_count.

To view the created top_3_loan:

```
db2 => select * from top_3_loan

BOOK_ID BOOK_TITLE                                         RENTAL_COUNT
------- -------------------------------------------------- ------------
CS1     Programming Fundamentals                                      4
CS2     The Future of Computer-Assisted Education                     2
IT1     Textbook of Information Technology                            2

  3 record(s) selected.
```

iii. Triggers

(a) Trigger to ensure records of bookdetail is parallel with the book_amt

```
CREATE TRIGGER check_bookdetail_data
BEFORE INSERT ON bookdetail
REFERENCING NEW AS new_row
FOR EACH ROW
BEGIN
    DECLARE book_amt INT;
    SET book_amt = (SELECT book_amt FROM book WHERE book_id =
new_row.book_id);

    IF book_amt = (SELECT COUNT(*) FROM bookdetail WHERE book_id =
new_row.book_id) THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot insert
more records into bookdetail for this book_id';
    END IF;
END
```

By comparing the count of existing records in the "bookdetail" table with the "book_amt" value, the trigger guarantees that the number of records does not exceed the specified limit. If the counts do not match, it raises an exception, preventing the insertion of additional records and maintaining the parallelism between "bookdetail" and "book_amt".

Results:

Book & bookdetail table before trigger:

```
db2 => select * from book

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
------- ------ ------ -------- ---------------------------------------------------- -----------
CS1     C002   G1            3 Programming Fundamentals                                    2019
CS2     C002   G2            2 The Future of Computer-Assisted Education                   2015
CS3     C002   R1            1 Computer Architecture                                       2019
IT1     C001   B1            2 Textbook of Information Technology                          1988
LAW1    C003   Y1            3 Cambridge Law Journal                                       2021
MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                       2021
CS4     C002   G1            1 Programming Fundamentals                                    2021
CS5     C002   B1            1 Data Science for Beginners                                  2009

  8 record(s) selected.
```

```
db2 => select  *from bookdetail

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567

  15 record(s) selected.
```

After trigger is invoked:

```
db2 => insert into bookdetail(book_id, barcode) values ('CS5',82379940)
DB20000I  The SQL command completed successfully.
db2 => insert into bookdetail(book_id, barcode) values ('CS5',82374940)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "Cannot
insert more records into bookdetail for this book_id".  SQLSTATE=45000
db2 => select  * from bookdetail

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567
           16 CS5     82379940

  16 record(s) selected.
```

(b) Trigger that verifies the eligibility of a member to loan a book

```
CREATE TRIGGER LoanEligibility
BEFORE INSERT ON Loan
REFERENCING NEW AS new
FOR EACH ROW mode db2sql
BEGIN
  DECLARE overdueFines DECIMAL(5,2);
  DECLARE memberStatus VARCHAR(10);
  DECLARE errorMessage VARCHAR(1000);

  SELECT COALESCE(SUM(fine_amt), 0)
  INTO overdueFines
  FROM Fine
  JOIN loandetail ld ON ld.loandetail_id = fine.loandetail_id
  JOIN loan l ON l.loan_id = ld.loan_id
  WHERE l.memb_id = new.memb_id
  AND fine.payment_date IS NULL;

  SELECT memb_status INTO memberStatus
  FROM Member
  WHERE memb_id = new.memb_id;

  IF overdueFines > 0 THEN
    SET errorMessage = 'You have outstanding fines: RM' ||
CAST(overdueFines AS VARCHAR(10)) || ', settle them before making new
loan.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = errorMessage;
  END IF;

  IF memberStatus <> 'Active' THEN
    SET errorMessage = 'Your member status: ' || memberStatus || ',
contact library for further assistance.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = errorMessage;
  END IF;
END
```

The trigger named LoanEligibility is triggered before the insertion of a new row into the Loan table. Three variables are declared:

1. overdueFines : To store the sum of the fine amounts of the member who wishes to loan a book.
2. memberStatus : To store the member status of the member who wishes to loan a book.

3. errorMessage : To hold the error message if any conditions are met.

The first SELECT statement calculates the sum of the fine amounts from the Fine table for the given member_ID (new.memb_id) who is trying to loan a book. If the member has any outstanding fines, message 'You have outstanding fines: RMxx , settle them before making new loan.' will be raised and stopped the member from loaning.
The second SELECT statement retrieves the membership status (memb_status) from the Member table for the given member ID (new.memb_id). If the member is not active, message 'Your member status: xxxxxxxxx , contact library for further assistance.' will be raised and stop the member from loaning.

To invoke the trigger of having outstanding fines:
insert into loan values('L007','M001',current date,'No')

Table before trigger invoke:

```
db2 => select  * from fine

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------
F1001   LD003         05/22/2023     1.00 05/22/2023
F1002   LD008         06/14/2023     2.00 06/15/2023
F1004   LD011         06/19/2023     1.00 -

  3 record(s) selected.

db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 No

  8 record(s) selected.

db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 -

  11 record(s) selected.
```

Message raised while trigger invoked:

```
db2 => insert into loan values('L007','M001',current date,'No')
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "You have
outstanding fines: RM1.00, settle them before making new loan".
SQLSTATE=45000
```

To invoke the trigger of member Status that was not Active:
insert into loan values('L007','M007',current date,'No')

Member table (to display member with their member status):

```
db2 => select * from member

MEMB_ID MEMB_NAME            MEMB_STATUS MEMB_TYPE
------- -------------------- ----------- ----------
M001    Yun Shi              Active      Student
M002    Qi Tong              Active      Student
M003    Jin Nan              Active      Student
M004    Kar Kin              Inactive    Student
M005    Joshua               Inactive    Lecturer
M006    Lily                 Active      Lecturer
M007    Kai Sheng            Inactive    Lecturer

  7 record(s) selected.
```

Message raised while trigger invoked:

```
db2 => insert into loan values('L007','M007',current date,'No')
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "Your
member status: Inactive, contact library for further assistance.".
SQLSTATE=45000
```

(c) Trigger to check whether a book is available for loaning

```
CREATE TRIGGER loan_available
BEFORE INSERT ON loandetail
REFERENCING NEW AS new_row
FOR EACH ROW MODE DB2SQL
BEGIN
    DECLARE book_loan_count INT;
    DECLARE book_reservation_count INT;

    IF EXISTS (
        SELECT *
        FROM loandetail
        WHERE bookdetail_id = new_row.bookdetail_id
        AND return_date IS NULL
    )
    THEN
        SET book_loan_count = 1;
    ELSE
        SET book_loan_count = 0;
    END IF;

    IF EXISTS (
        SELECT *
        FROM reservation
        WHERE bookdetail_id = new_row.bookdetail_id
        AND reserve_status = 'Pending'
    )
    THEN
        SET book_reservation_count = 1;
    ELSE
        SET book_reservation_count = 0;
    END IF;

    IF book_reservation_count > 0 THEN
        SIGNAL SQLSTATE '75001' SET MESSAGE_TEXT = 'The book is
currently being reserved.';
    ELSEIF book_loan_count > 0 THEN
        SIGNAL SQLSTATE '75001' SET MESSAGE_TEXT = 'The book is
currently loaned out.';
    END IF;
END
```

This trigger ensures that a new row can only be inserted into the loandetail table if the book is neither loaned out nor being reserved.

1. Two variables are declared: book_loan_count and book_reservation_count to keep track of the counts of existing loan records and reservation records for the book being inserted.
2. The trigger then checks if there are any active loan records for the same bookdetail_id (indicating that the book is currently loaned out). If such records exist, it sets book_loan_count to 1; otherwise, it sets it to 0.
3. Next, the trigger checks if there are any pending reservation records for the same bookdetail_id (indicating that the book is currently being reserved). If such records exist, it sets book_reservation_count to 1; otherwise, it sets it to 0.
4. Based on the values of book_loan_count and book_reservation_count, the trigger performs the following actions:
    a. If book_reservation_count is greater than 0, it raises an error by signaling SQLSTATE '75001' with a custom error message stating that the book is currently being reserved.
    b. If book_loan_count is greater than 0, it raises an error by signaling SQLSTATE '75001' with a custom error message stating that the book is currently loaned out.

Table of loandetail & reservation:

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 -

  11 record(s) selected.

db2 => select * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------
M001    04/23/2023               2 Expired
M006    04/23/2023               9 Completed
M006    04/30/2023               6 Completed
M003    05/07/2023              10 Expired
M005    05/23/2023              11 Expired
M006    06/01/2023               1 Completed
M001    06/13/2023               4 Expired
M001    06/17/2023               4 Pending

  8 record(s) selected.
```

Results after trigger is invoked:
Loaning not available due to book is being reserved

```
db2 => insert into loandetail values ('LD009','L004',4, NULL)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "The book
is currently being reserved.".  SQLSTATE=75001
```

Loaning not available due to book is currently loaned out (have not been returned)

```
db2 => insert into loandetail values('LD010','L004',3,NULL)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "The book
is currently loaned out.".  SQLSTATE=75001
```

(d) Trigger to check whether a book is available for reservation

```
CREATE TRIGGER rsv_available
BEFORE INSERT ON reservation
REFERENCING NEW AS new_row
FOR EACH ROW MODE DB2SQL
BEGIN
    DECLARE book_loan_count INT;
    DECLARE book_reservation_count INT;

    IF EXISTS (
        SELECT 1
        FROM loandetail
        WHERE bookdetail_id = new_row.bookdetail_id
    )
    THEN
        SELECT COUNT(*) INTO book_loan_count
        FROM loan l
        JOIN loandetail ld ON l.loan_id = ld.loan_id
        WHERE ld.bookdetail_id = new_row.bookdetail_id
        AND ld.return_date IS NULL;
    ELSE
        SET book_loan_count = 0;
    END IF;

    IF EXISTS (
        SELECT 1
        FROM reservation
        WHERE bookdetail_id = new_row.bookdetail_id
    )
    THEN
        SELECT COUNT(*) INTO book_reservation_count
        FROM reservation r
        WHERE r.bookdetail_id = new_row.bookdetail_id
        AND r.reserve_status ='Pending';
    ELSE
        SET book_reservation_count = 0;
    END IF;

    IF book_loan_count > 0 THEN
        SIGNAL SQLSTATE '75001' SET MESSAGE_TEXT = 'The book is
currently loaned out.';
```

```
    ELSEIF book_reservation_count > 0 THEN
        SIGNAL SQLSTATE '75001' SET MESSAGE_TEXT = 'The book is
currently being reserved.';
    END IF;
END
```

This trigger ensures that a new row can only be inserted into the reservation table if the book is not loaned out or being reserved.

1. Two variables are declared: book_loan_count and book_reservation_count to keep track of the counts of existing loan records and reservation records for the book being inserted.
2. The trigger then checks if there are any existing loan records for the same bookdetail_id in the loandetail table. If such records exist, it retrieves the count of active loan records (where the return_date is NULL) and assigns it to book_loan_count. Otherwise, it sets book_loan_count to 0.
3. Next, the trigger checks if there are any existing reservation records for the same bookdetail_id in the reservation table. If such records exist, it retrieves the count of pending reservation records (where the reserve_status is 'Pending') and assigns it to book_reservation_count. Otherwise, it sets book_reservation_count to 0.
4. Based on the values of book_loan_count and book_reservation_count, the trigger performs the following actions:
    a. If book_loan_count is greater than 0, it raises an error by signaling SQLSTATE '75001' with a custom error message stating that the book is currently loaned out.
    b. If book_reservation_count is greater than 0, it raises an error by signaling SQLSTATE '75001' with a custom error message stating that the book is currently being reserved.

Table of loandetail & reservation:

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 -

  11 record(s) selected.

db2 => select * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------
M001    04/23/2023               2 Expired
M006    04/23/2023               9 Completed
M006    04/30/2023               6 Completed
M003    05/07/2023              10 Expired
M005    05/23/2023              11 Expired
M006    06/01/2023               1 Completed
M001    06/13/2023               4 Expired
M001    06/17/2023               4 Pending

  8 record(s) selected.
```

Results after trigger is invoked:

Book not available for reservation as the book is currently reserved

```
db2 => insert into reservation values('M002',current date, 4, default)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "The book
is currently being reserved.".  SQLSTATE=75001
```

Book not available for reservation as the book is currently loaned out

```
db2 => insert into reservation values('M002', current date, 3, default)
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "The book
is currently loaned out.".  SQLSTATE=75001
```

Hence, no new records will be entered into the reservation table

```
db2 => select  * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------
M001    04/23/2023               2 Expired
M006    04/23/2023               9 Completed
M006    04/30/2023               6 Completed
M003    05/07/2023              10 Expired
M005    05/23/2023              11 Expired
M006    06/01/2023               1 Completed
M001    06/13/2023               4 Expired
M001    06/17/2023               4 Pending

  8 record(s) selected.
```

(e) To generate fine_id automatically

```
CREATE TRIGGER autoFineId
BEFORE INSERT ON fine
REFERENCING NEW AS new
FOR EACH ROW
WHEN (new.fine_id IS NULL)
BEGIN ATOMIC
    SET new.fine_id = CONCAT('F', CHAR(next value for fine_id_seq));
END
```

The trigger autoFineId created is to generate fine_id automatically before every new row being inserted in the fine table. Condition was set such that if the fine_id is null, it will assign a new fine_id to every data which start from F together with an integer that has been declared in a created sequence earlier.

This trigger is mainly executed while the procedure calculateFine() is being called.
To invoke trigger:

```
Insert into fine (loandetail_id, fine_date, fine_amt, payment_date)
values ('LD011', '2023-06-19',1,null)
```

Result:

```
db2 => INSERT INTO FINE(loandetail_id, fine_date, fine_amt, payment_date) VALUES ('LD011','2023-06-19',
1, null)
DB20000I  The SQL command completed successfully.
db2 => SELECT * FROM FINE

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------
F1000   LD003         05/22/2023     1.00 05/22/2023
F1001   LD008         06/14/2023     2.00 06/15/2023
F1002   LD011         06/19/2023     1.00 -

  3 record(s) selected.
```

(f) To update return status of each loan

```
CREATE TRIGGER triggerReturn
AFTER UPDATE ON loandetail
REFERENCING OLD AS old_row
FOR EACH ROW
MODE DB2SQL
BEGIN
  UPDATE loan SET return_status = 'Yes' WHERE return_status IS NOT
NULL AND old_row.loan_id = loan.loan_id;
END
```

Trigger created named triggerReturn is to update the return status to yes when the return_date of loandetail table has a value.

To invoke the trigger:

```
update loandetail set return_date = current date where
loandetail_id='LD0011'
```

Loan and loanDetail table before trigger:

```
db2 => select  * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 No
L009    M002    06/15/2023 Yes
```

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 -
LD012         L009                1 06/19/2023
```

Results after trigger is invoked:

```
db2 => update loandetail set return_date=current date where loandetail_id='LD011'
DB20000I  The SQL command completed successfully.
db2 => select  * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 06/19/2023
LD012         L009                1 06/19/2023

  12 record(s) selected.
```

```
db2 => select  * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 Yes
L009    M002    06/15/2023 Yes

  9 record(s) selected.
```

(g) Trigger to count the maximum book a member can loan

```
CREATE TRIGGER loanlimit
BEFORE INSERT ON loan
REFERENCING NEW AS new_row
FOR EACH ROW BEGIN DECLARE total_loans INT;
DECLARE member_type VARCHAR(10);
SELECT memb_type
INTO member_type
FROM member WHERE memb_id = new_row.memb_id;
 SELECT COUNT(*)
INTO total_loans
FROM loandetail ld
JOIN loan l ON ld.loan_id = l.loan_id
WHERE l.memb_id = new_row.memb_id
AND l.return_status = 'No'; IF (member_type = 'Student' AND
total_loans >= 2) OR (member_type = 'Lecturer' AND total_loans >= 10)
THEN SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Loan limit exceeded for the borrower.';
END IF;
END
```

Trigger named as loanlimit is to check/count the maximum book a member can loan. One student can loan a maximum of 2 books while a lecturer can loan maximum10 books.

To invoke trigger:

```
db2 => INSERT INTO loan VALUES ('L0017', 'M003', '2023-6-19', 'No')
```

Result:

```
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0438N  Application raised error or warning with diagnostic text: "Loan
limit exceeded for the borrower.".  SQLSTATE=45000
```

iv. Stored procedure

(a) Stored procedure to search for books by keyword, book title or author

```
CREATE PROCEDURE searchBook(IN searchType VARCHAR(10), IN searchValue
VARCHAR(20))
BEGIN
  DECLARE c1 CURSOR WITH RETURN FOR
    SELECT b.book_id, b.cat_id, b.tag_id, b.book_amt, b.book_title,
b.pub_year
    FROM book b
    WHERE b.book_id IN (
      SELECT bc.book_id
      FROM category c, book bc
      WHERE c.cat_id = bc.cat_id
        AND UPPER(c.cat_keyword) = UPPER(searchValue)
    )
    ORDER BY b.book_title;

  DECLARE c2 CURSOR WITH RETURN FOR
    SELECT b.book_id, b.cat_id, b.tag_id, b.book_amt, b.book_title,
b.pub_year
    FROM book b
    INNER JOIN bookauthor ba ON b.book_id = ba.book_id
    INNER JOIN author a ON ba.auth_id = a.auth_id
    WHERE UPPER(a.fname) = UPPER(searchValue)
      OR UPPER(a.lname) = UPPER(searchValue)
      OR UPPER(a.fname || ' ' || a.lname) = UPPER(searchValue)
    ORDER BY b.book_title;

  DECLARE c3 CURSOR WITH RETURN FOR
    SELECT b.book_id, b.cat_id, b.tag_id, b.book_amt, b.book_title,
b.pub_year
    FROM book b
    WHERE UPPER(b.book_title) LIKE '%' || UPPER(searchValue) || '%'
    ORDER BY b.book_title;

  IF UPPER(searchType) = 'KEYWORD' THEN
    OPEN c1;
  ELSEIF UPPER(searchType) = 'AUTHOR' THEN
    OPEN c2;
```

```
  ELSEIF UPPER(searchType) = 'TITLE' THEN
    OPEN c3;
  ELSE
    SIGNAL SQLSTATE '38000' SET MESSAGE_TEXT = 'Invalid search type
provided. Please enter KEYWORD, AUTHOR, or TITLE';
  END IF;
END
```

This stored procedure **searchBook** allows searching of books based on different criteria: keyword, author or title. User will have to specify the 'searchType' (criteria) that they want, followed by the 'searchValue'(value to search for). All of the user input are set to uppercase with the UPPER() function so that the search is not case-sensitive. Then 3 cursors are declared; c1 cursor will retrieve books that match the 'searchValue' as keyword, c2 cursor retrieves books that match the 'searchValue' as author while c3 cursor retrieves books that match the 'searchValue' as book title. After that, the if-else statement will check the 'searchType'; if 'searchType' is 'KEYWORD', c1 will be opened, if it's 'AUTHOR' c2 will be opened, if it's 'TITLE' c3 will be opened, and if it's not any of these, it indicates an invalid search type is entered with a message text showing up. The output table will display data in an ascending order of book_title.

To invoke the stored procedure:

```
Call searchBook('keyword','cs')
Call searchBook('title','computer')
Call searchBook('author','winson lim')
```

Searching of books by keyword:

```
db2 => Call searchBook('keyword','cs')


  Result set 1
  --------------

  BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
  ------- ------ ------ -------- ------------------------------------------------- -----------
  CS3     C002   R1            1 Computer Architecture                                    2019
  CS5     C002   B1            1 Data Science for Beginners                               2009
  CS1     C002   G1            3 Programming Fundamentals                                 2019
  CS4     C002   G1            1 Programming Fundamentals                                 2021
  CS2     C002   G2            2 The Future of Computer-Assisted Education                2015

  5 record(s) selected.

  Return Status = 0
```

Searching of books by book author (user can either enter author's full name or author's first name or author's last name):

```
db2 => Call searchBook('author','winson')


  Result set 1
  --------------

  BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
  ------- ------ ------ -------- ------------------------------------------------- -----------
  CS3     C002   R1            1 Computer Architecture                                    2019
  CS5     C002   B1            1 Data Science for Beginners                               2009
  MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                    2021
  CS1     C002   G1            3 Programming Fundamentals                                 2019

  4 record(s) selected.

  Return Status = 0
db2 => Call searchBook('author','winson lim')


  Result set 1
  --------------

  BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
  ------- ------ ------ -------- ------------------------------------------------- -----------
  CS3     C002   R1            1 Computer Architecture                                    2019
  CS5     C002   B1            1 Data Science for Beginners                               2009
  MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                    2021
  CS1     C002   G1            3 Programming Fundamentals                                 2019

  4 record(s) selected.

  Return Status = 0
```

Searching of books by book title:

```
db2 => Call searchBook('title','computer')


  Result set 1
  --------------

  BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
  ------- ------ ------ -------- ------------------------------------------------- -----------
  CS3     C002   R1            1 Computer Architecture                                    2019
  CS2     C002   G2            2 The Future of Computer-Assisted Education                2015

  2 record(s) selected.

  Return Status = 0
```

(b) Stored procedure to automatically update return_status and return_date

```
CREATE PROCEDURE updateReturn(IN ld_id VARCHAR(5))
BEGIN
  DECLARE c CURSOR WITH RETURN FOR
  SELECT * FROM loandetail join loan on loandetail.loan_id =
loan.loan_id;

  UPDATE loandetail
  SET return_date = CURRENT DATE
  WHERE loandetail_id = ld_id;

  UPDATE loan
  SET return_status = 'Yes'
  WHERE loan_id = (SELECT loan_id FROM loandetail WHERE loandetail_id
= ld_id);

  OPEN c;
END
```

The stored procedure named updateReturn takes one input parameter ld_id which is the loandetail_id of an individual who returned the book at the moment.
The first UPDATE statement updates the return_date column of the loandetail table to the current date when an individual returns the book.
The second UPDATE statement updates the return_status column of the loan table to 'Yes' after the first update statement being executed to the particular individual.
updateReturn will return the latest information of both loan and loanDetail tables.

To execute the updateReturn procedure:

```
call updateReturn('LD006')
```

Table of loan & loandetail before procedure is executed:

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 -
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 06/19/2023
LD012         L009                1 06/19/2023

  12 record(s) selected.

db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 No
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 Yes
L009    M002    06/15/2023 Yes

  9 record(s) selected.
```

Results after procedure is executed:

```
db2 => call updateReturn('LD006')


  Result set 1
  --------------

  LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
  ------------- ------- ------------- ----------- ------- ------- ---------- -------------
  LD001         L001                9 05/01/2023  L001    M006    04/25/2023 Yes
  LD002         L002                6 05/10/2023  L002    M006    05/01/2023 Yes
  LD003         L003                1 05/22/2023  L003    M003    05/20/2023 Yes
  LD004         L004                1 06/03/2023  L004    M006    06/02/2023 Yes
  LD005         L005                2 06/10/2023  L005    M001    06/10/2023 Yes
  LD006         L005                6 06/19/2023  L005    M001    06/10/2023 Yes
  LD007         L006                4 06/12/2023  L006    M002    06/11/2023 Yes
  LD008         L006                4 06/15/2023  L006    M002    06/11/2023 Yes
  LD009         L008                7 06/18/2023  L008    M001    06/17/2023 Yes
  LD011         L008                3 06/19/2023  L008    M001    06/17/2023 Yes
  LD010         L008                8 06/19/2023  L008    M001    06/17/2023 Yes
  LD012         L009                1 06/19/2023  L009    M002    06/15/2023 Yes

  12 record(s) selected.
```

Changes in loan and loandetail table after procedure is executed:

```
db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M006    04/25/2023 Yes
L002    M006    05/01/2023 Yes
L003    M003    05/20/2023 Yes
L004    M006    06/02/2023 Yes
L005    M001    06/10/2023 Yes
L006    M002    06/11/2023 Yes
L007    M006    06/12/2023 Yes
L008    M001    06/17/2023 Yes
L009    M002    06/15/2023 Yes

  9 record(s) selected.
```

```
db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L001                9 05/01/2023
LD002         L002                6 05/10/2023
LD003         L003                1 05/22/2023
LD004         L004                1 06/03/2023
LD005         L005                2 06/10/2023
LD006         L005                6 06/19/2023
LD007         L006                4 06/12/2023
LD008         L006                4 06/15/2023
LD009         L008                7 06/18/2023
LD010         L008                8 06/19/2023
LD011         L008                3 06/19/2023
LD012         L009                1 06/19/2023

  12 record(s) selected.
```

(c) To check the expiry status of reservation

```
CREATE PROCEDURE updateExpiry
LANGUAGE SQL
BEGIN
    DECLARE c CURSOR WITH RETURN FOR
        SELECT * FROM reservation;

    FOR v_row AS
        SELECT * FROM reservation
    DO
        IF (v_row.reserve_status = 'Pending') THEN
            UPDATE reservation
            SET reserve_status = 'Expired'
            WHERE reserve_date < CURRENT DATE - 2 DAYS
                AND reserve_status = 'Pending';
        END IF;
    END FOR;

    OPEN c;
END
```

The procedure updateExpiry will check the expiry status of the reservation. It will be executed after calling the procedure.
It is designed to update the reserve_status column to 'Expired' for rows such that the reserve_date is more than 2 days older than the current date.
It will return the latest reservation table after calling the procedure.

To invoke the procedure:

```
Call updateExpiry
```

Table of reservation:

```
db2 => insert into reservation values('M005','2023-06-15',1,DEFAULT)
DB20000I  The SQL command completed successfully.
db2 => select * from reservation

MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
------- ------------ ------------- --------------
M001    04/23/2023               2 Expired
M006    04/23/2023               9 Completed
M006    04/30/2023               6 Completed
M003    05/07/2023              10 Expired
M005    05/23/2023              11 Expired
M006    06/01/2023               1 Completed
M001    06/13/2023               4 Expired
M001    06/17/2023               4 Pending
M005    06/15/2023               1 Pending

  9 record(s) selected.
```

Results after procedure is executed:

```
db2 => call updateExpiry


  Result set 1
  --------------

  MEMB_ID RESERVE_DATE BOOKDETAIL_ID RESERVE_STATUS
  ------- ------------ ------------- --------------
  M001    04/23/2023               2 Expired
  M006    04/23/2023               9 Completed
  M006    04/30/2023               6 Completed
  M003    05/07/2023              10 Expired
  M005    05/23/2023              11 Expired
  M006    06/01/2023               1 Completed
  M001    06/13/2023               4 Expired
  M001    06/17/2023               4 Pending
  M005    06/15/2023               1 Expired

  9 record(s) selected.

  Return Status = 0
```

(d) To update member status as suspended/active

```
CREATE PROCEDURE updateMem
BEGIN
    DECLARE v_memb_id VARCHAR(10);
    DECLARE v_payment_date DATE;

    FOR suspended_members AS
        SELECT m.memb_id, f.payment_date
        FROM member m
        JOIN loan l ON l.memb_id = m.memb_id
        JOIN loandetail ld ON ld.loan_id = l.loan_id
        JOIN fine f ON f.loandetail_id = ld.loandetail_id
        WHERE f.payment_date IS NULL
          OR ld.return_date IS NULL
    DO
        SET v_memb_id = suspended_members.memb_id;
        SET v_payment_date = suspended_members.payment_date;
        UPDATE member SET memb_status = 'Suspended' WHERE memb_id =
v_memb_id;
    END FOR;

    FOR active_members AS
        SELECT m.memb_id, f.payment_date
        FROM member m
        JOIN loan l ON l.memb_id = m.memb_id
        JOIN loandetail ld ON ld.loan_id = l.loan_id
        JOIN fine f ON f.loandetail_id = ld.loandetail_id
        WHERE f.payment_date IS NOT NULL
          OR ld.return_date IS NOT NULL
    DO
        SET v_memb_id = active_members.memb_id;
        SET v_payment_date = active_members.payment_date;
        UPDATE member SET memb_status = 'Active' WHERE memb_id =
v_memb_id;
    END FOR;

END
```

Procedure created named updateMem is to update the member status when being called. The member status(memb_status) of the member who is being fined and has not paid will be changed

to ‘Suspended”. When there is a payment_date of a member at fine table, the member_status of the member who has paid the fine amount will be revived as ‘Active’.

Below are fine, loanDetail and loan table to show member who did not return book:

```
db2 => select * from fine

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE PAYMENT_AMT
------- ------------- ---------- -------- ------------ -----------
F1000   LD001         06/17/2023     3.00 -                      -
F1001   LD005         06/15/2023     5.00 -                      -
F1002   LD006         05/03/2023    48.00 -                      -
F1003   LD007         06/15/2023     5.00 -                      -
F1004   LD013         06/17/2023     3.00 -                      -

  5 record(s) selected.

db2 => select * from loandetail

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L002                1 -
LD002         L001                2 06/13/2023
LD003         L003                9 06/13/2023
LD004         L004                5 06/18/2023
LD005         L005                3 -
LD006         L006                1 -
LD007         L005                4 -
LD012         L009                1 06/19/2023
LD013         L0010               1 -

  9 record(s) selected.

db2 => select * from loan

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M001    05/01/2023 Yes
L002    M004    06/15/2023 No
L003    M003    06/12/2023 Yes
L004    M002    05/01/2023 Yes
L005    M001    06/13/2023 No
L006    M007    05/01/2023 No
L0010   M002    06/15/2023 Yes
L009    M002    06/15/2023 Yes
```

For those member who did not return book, their member status has changed to suspended by calling the procedure updatemem

```
db2 => call updatemem

  Return Status = 0
db2 => select * from member

MEMB_ID MEMB_NAME           MEMB_STATUS MEMB_TYPE
------- ------------------- ----------- ----------
M001    Yun Shi             Suspended   Student
M002    Qi Tong             Suspended   Student
M003    Jin Nan             Active      Student
M005    Joshua              Inactive    Lecturer
M006    Linda               Active      Lecturer
M007    Lisa                Suspended   Lecturer

  6 record(s) selected.
```

Changes in member status when they return their loan:

```
db2 => update loandetail set return_date = current date where loandetail_id = 'LD006'
DB20000I  The SQL command completed successfully.
db2 => call updatemem

  Return Status = 0
```

```
db2 => select * from member

MEMB_ID MEMB_NAME           MEMB_STATUS MEMB_TYPE
------- ------------------- ----------- ----------
M001    Yun Shi             Suspended   Student
M002    Qi Tong             Suspended   Student
M003    Jin Nan             Active      Student
M005    Joshua              Inactive    Lecturer
M006    Linda               Active      Lecturer
M007    Lisa                Suspended   Lecturer
```

v. View

(a) To view book that is in pending reservation status

```
CREATE VIEW bookStatus AS
(SELECT bookdetail.bookdetail_id, book.book_title
FROM book, bookdetail, reservation
WHERE bookdetail.book_id = book.book_id
AND bookdetail.bookdetail_id = reservation.bookdetail_id
AND reservation.reserve_status = 'Pending')
```

The view bookStatus created is to list out all of the bookdetail_id and book_title columns from the bookDetail and book tables based on the condition that reserve_status of reservation table is under 'Pending'.

To view the created bookStatus:

```
SELECT * from bookStatus
```

Results:

```
db2 => select  * from bookStatus

BOOKDETAIL_ID BOOK_TITLE
------------- ----------------------------------------------------
            4 The Future of Computer-Assisted Education

  1 record(s) selected.
```

vi. Subqueries/nested queries

(a) To display total fine amount and unsettled fine amount of each member

```
SELECT
    m.memb_id,
    m.memb_name,
    COALESCE(SUM(f.fine_amt), 0) AS total_fine_amt,
    COALESCE((SELECT SUM(fine_amt) FROM fine WHERE payment_date IS
NULL AND loandetail_id IN (SELECT loandetail_id FROM loandetail WHERE
loan_id IN (SELECT loan_id FROM loan WHERE memb_id = m.memb_id))), 0)
AS total_unpaid_fine_amt
FROM member m
LEFT JOIN loan l ON m.memb_id = l.memb_id
LEFT JOIN loandetail ld ON l.loan_id = ld.loan_id
LEFT JOIN fine f ON ld.loandetail_id = f.loandetail_id
GROUP BY m.memb_id, m.memb_name
```

The purpose of this query is to get information about each member, including their IDs, names, total fine amounts, and total unpaid fine amounts.

The COALESCE function is used to handle null results from the subquery, and it will replace any null value with 0.
The subquery starts from (SELECT SUM(fine_amt) FROM fine …) that calculates the sum of fine_amt from the fine table.
From inner to outer query:

1. (SELECT loan_id FROM loan WHERE memb_id = m.memb_id)
2. (SELECT loandetail_id FROM loandetail WHERE loan_id IN (SELECT loan_id FROM loan WHERE memb_id = m.memb_id))
3. (SELECT SUM(fine_amt) FROM fine WHERE payment_date IS NULL AND loandetail_id IN (SELECT loandetail_id FROM loandetail WHERE loan_id IN (SELECT loan_id FROM loan WHERE memb_id = m.memb_id)))

Results:

```
db2 => SELECT m.memb_id, m.memb_name, COALESCE(SUM(f.fine_amt), 0) AS total_fine_amt, COALESCE((SELECT SUM(fine_amt) FROM f
ine WHERE payment_date IS NULL AND loandetail_id IN (SELECT loandetail_id FROM loandetail WHERE loan_id IN (SELECT loan_id
FROM loan WHERE memb_id = m.memb_id))), 0) AS total_unpaid_fine_amt FROM member m LEFT JOIN loan l ON m.memb_id = l.memb_id
 LEFT JOIN loandetail ld ON l.loan_id = ld.loan_id LEFT JOIN fine f ON ld.loandetail_id = f.loandetail_id GROUP BY m.memb_i
d, m.memb_name

MEMB_ID MEMB_NAME           TOTAL_FINE_AMT                TOTAL_UNPAID_FINE_AMT
------- ------------------- ----------------------------- -----------------------------
M001    Yun Shi                                    1.00                          1.00
M002    Qi Tong                                    5.00                          0.00
M003    Jin Nan                                    1.00                          0.00
M004    Kar Kin                                    0.00                          0.00
M005    Joshua                                     0.00                          0.00
M006    Lily                                       0.00                          0.00
M007    Kai Sheng                                  0.00                          0.00

  7 record(s) selected.
```

vii. At least *four* queries not covered in lecture/tutorial

(a) Using **while loop** in a stored procedure

```
CREATE PROCEDURE insert_bookdetail(IN book_id VARCHAR(5), IN book_amt
SMALLINT)
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE temp_barcode VARCHAR(10);
    DECLARE barcode_exists INT;

    WHILE i <= book_amt DO
        SET temp_barcode = LPAD(VARCHAR(INTEGER(RAND() * 100000000)),
8, '0');
        SET barcode_exists = (SELECT COUNT(*) FROM bookdetail WHERE
barcode = temp_barcode);

        WHILE barcode_exists > 0 DO
            SET temp_barcode = LPAD(VARCHAR(INTEGER(RAND() *
100000000)), 8, '0');
            SET barcode_exists = (SELECT COUNT(*) FROM bookdetail
WHERE barcode = temp_barcode);
        END WHILE;

        INSERT INTO bookdetail (book_id, barcode) VALUES (book_id,
temp_barcode);
        SET i = i + 1;
    END WHILE;
END
```

The purpose of this stored procedure is to insert multiple bookdetails records into the bookdetail table based on its book_amt in the book table.

The procedure will be executed repeatedly for a specified number of iterations based on the value of the variable "book_amt". Within each iteration, it generates a random barcode ("temp_barcode") using the **RAND()** function. It then checks if the generated barcode already exists in the bookdetail table. If the generated barcode already exists, it repeats the barcode generation process until a unique barcode is obtained. Once a unique barcode is generated, it performs an INSERT operation into the bookdetail table, associating it with the specified book_id.

The loop continues until the desired number of book details has been inserted. After the loop completes, the procedure finishes its execution.

To execute the stored procedure:

```
insert into book values ('', 'C005', 'Y2', 3, 'The Little Prince',
2019)
call insert_bookdetail('FIC1',3)
```

Tables of book & bookdetail before procedure is executed:

```
db2 => select * from book

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
------- ------ ------ -------- ------------------------------------------------- -----------
CS1     C002   G1            3 Programming Fundamentals                                  2019
CS2     C002   G2            2 The Future of Computer-Assisted Education                 2015
CS3     C002   R1            1 Computer Architecture                                     2019
IT1     C001   B1            2 Textbook of Information Technology                        1988
LAW1    C003   Y1            3 Cambridge Law Journal                                     2021
MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                     2021
CS4     C002   G1            1 Programming Fundamentals                                  2021
CS5     C002   B1            1 Data Science for Beginners                                2009

  8 record(s) selected.

db2 => select * from bookdetail

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567
           16 CS5     82379940
```

To execute the procedure:
insert into book values ('', 'C005', 'Y2', 3, 'The Little Prince', 2019)
insert into book values ('', 'C005', 'Y2', 2, 'The Little Prince', 2019)
insert into book values ('', 'C005', 'Y2', 3, 'The Little Prince', 2019)
call insert_bookdetail('FIC1',3)
call insert_bookdetail('FIC2',2)
call insert_bookdetail('FIC3',3)

Results:

```
db2 => SELECT * FROM BOOK

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
------- ------ ------ -------- ------------------------------------------------- -----------
CS1     C002   G1            3 Programming Fundamentals                                  2019
CS2     C002   G2            2 The Future of Computer-Assisted Education                 2015
CS3     C002   R1            1 Computer Architecture                                     2019
IT1     C001   B1            2 Textbook of Information Technology                        1988
LAW1    C003   Y1            3 Cambridge Law Journal                                     2021
MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                     2021
CS4     C002   G1            1 Programming Fundamentals                                  2021
CS5     C002   B1            1 Data Science for Beginners                                2009
FIC1    C005   Y2            3 The Little Prince                                         2019
FIC2    C005   Y2            3 The Sun                                                   2019
FIC3    C005   Y2            3 The Flower                                                2019

  11 record(s) selected.
```

```
db2 => call insert_bookdetail('FIC1',3)

  Return Status = 0
db2 => call insert_bookdetail('FIC2',2)

  Return Status = 0
db2 => call insert_bookdetail('FIC3',3)

  Return Status = 0
db2 => SELECT * FROM BOOKDETAIL

BOOKDETAIL_ID BOOK_ID BARCODE
------------- ------- ----------
            1 CS1     63527777
            2 CS1     87984797
            3 CS1     43432424
            4 CS2     98991837
            5 CS2     12998467
            6 CS3     21338467
            7 IT1     78772364
            8 IT1     12378849
            9 LAW1    65364588
           10 LAW1    91127463
           11 LAW1    87366577
           12 MT1     65337489
           13 MT1     39001176
           14 MT1     26677784
           15 CS4     21338567
           16 CS5     82379940
           17 FIC1    79082613
           18 FIC1    32917264
           19 FIC1    54332102
           20 FIC2    17059846
           21 FIC2    70769371
           22 FIC3    65556810
           23 FIC3    18668172
           24 FIC3    81215857

  24 record(s) selected.
```

(b) To generate sequence starting from 1000 (for the fine_id generation)

```
CREATE SEQUENCE fine_id_seq AS INTEGER START WITH 1000 INCREMENT BY 1
```

The **sequence** fine_id_seq created will start with the value 1000 and increment by 1 each time. The values created by the sequence will be used in the trigger autoFineId with the concatenation of 'F' to generate the fine_id.

```
db2 => CREATE SEQUENCE fine_id_seq AS INTEGER START WITH 1000 INCREMENT BY 1
DB20000I  The SQL command completed successfully.
```

```
db2 => SELECT * FROM FINE

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------
F1000   LD003         05/22/2023     1.00 05/22/2023
F1001   LD008         06/14/2023     2.00 06/15/2023

  2 record(s) selected.
```

After insertion:

```
db2 => INSERT INTO FINE(loandetail_id, fine_date, fine_amt, payment_date) VALUES ('LD011','2023-06-19',
1, null)
DB20000I  The SQL command completed successfully.
db2 => SELECT * FROM FINE

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE
------- ------------- ---------- -------- ------------
F1000   LD003         05/22/2023     1.00 05/22/2023
F1001   LD008         06/14/2023     2.00 06/15/2023
F1002   LD011         06/19/2023     1.00 -

  3 record(s) selected.
```

(c) To insert data into fine table automatically

```
CREATE PROCEDURE calculateFine()
LANGUAGE SQL
BEGIN
    DECLARE v_loandetail_id VARCHAR(5);
    DECLARE v_loan_days INT;
    DECLARE v_fine_amt DECIMAL(5,2);
    DECLARE v_loan_period INT;
    DECLARE v_fine_date DATE;

    DECLARE c CURSOR FOR
        SELECT ld.loandetail_id, DAYS(CURRENT DATE) -
DAYS(l.loan_date), t.loan_period
        FROM loandetail ld
        JOIN loan l ON ld.loan_id = l.loan_id
        JOIN bookdetail bd ON ld.bookdetail_id = bd.bookdetail_id
        JOIN book b ON b.book_id = bd.book_id
        JOIN colortag t ON t.tag_id = b.tag_id
        WHERE ld.return_date IS NULL;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_loandetail_id =
NULL;

    OPEN c;

    fetch_loop: LOOP
        FETCH c INTO v_loandetail_id, v_days_fine, v_loan_period;

        IF v_loandetail_id IS NULL THEN
            LEAVE fetch_loop;
        END IF;

        IF v_loan_days > v_loan_period THEN
            SET v_fine_amt = (v_loan_days - v_loan_period);
            SET v_fine_date = DATE(DAYS((SELECT loan_date FROM loan l
JOIN loandetail ld ON ld.loan_id = l.loan_id WHERE ld.loandetail_id =
v_loandetail_id)) + v_loan_period + 1);
        ELSE
            SET v_fine_amt = 0;
            SET v_fine_date = NULL;
```

```
        END IF;

        MERGE INTO fine f
        USING (
            SELECT v_loandetail_id, v_fine_date, v_fine_amt
            FROM loandetail ld
            WHERE ld.loandetail_id = v_loandetail_id
        ) AS s (loandetail_id, fine_date, fine_amt)
        ON f.loandetail_id = s.loandetail_id
        WHEN MATCHED THEN
            UPDATE SET f.fine_date = s.fine_date, f.fine_amt =
s.fine_amt
        WHEN NOT MATCHED THEN
            INSERT (loandetail_id, fine_date, fine_amt) VALUES
(s.loandetail_id, s.fine_date, s.fine_amt);

    END LOOP fetch_loop;

    CLOSE c;
END
```

Trigger calculateFine() is to insert and update data in the fine table automatically just by calling this procedure.
Five variables have been declared within the procedure to store needed information which is:

1. v_loandetail_id : to store the loandetail id of members that do not return book
2. v_loan_days : to store the number of days of a book that has been loaned
3. v_fine_amt : to store the fine amount of a loan
4. v_loan_period : to get the loan period of each book loaned
5. v_fine_date : to calculate and get the first day of being fined

When the cursor is open, the fetch loop begins, which retrieves data from the cursor into the declared variables. The purpose of **DECLARE CONTINUE HANDLER FOR NOT FOUND** is to handle the situation when the cursor's FETCH statement fails to fetch a row, indicating that there are no more rows to process. In this case, the NOT FOUND condition is raised, and the handler is triggered. When the handler is triggered, it sets v_loandetail_id to NULL, indicating no more rows to fetch, the fetch loop is left. Condition was set such that if loan days is more than loan period, fine date and fine amount(fine_amt) will be calculated then insert into fine table if the loandetail_id does not exist in the fine table, else data will be updated into fine table according to their loandetail_id.

Loan, loanDetail and fine table (to show the loaning status of each member):

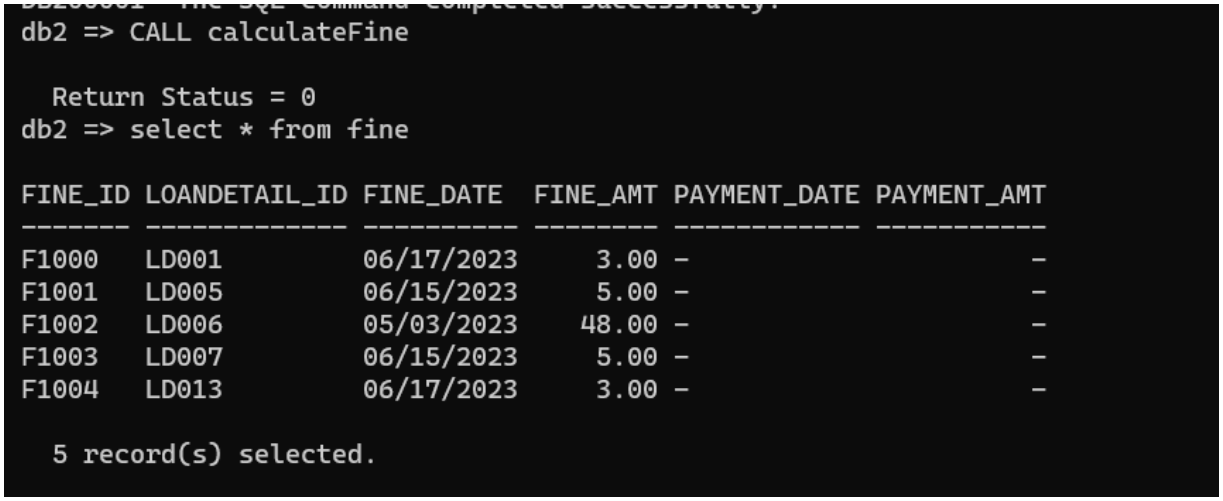
```
db2 => SELECT * FROM LOAN

LOAN_ID MEMB_ID LOAN_DATE  RETURN_STATUS
------- ------- ---------- -------------
L001    M001    05/01/2023 Yes
L002    M004    06/15/2023 No
L003    M003    06/12/2023 Yes
L004    M002    05/01/2023 Yes
L005    M001    06/13/2023 No
L006    M007    05/01/2023 No
L0010   M002    06/15/2023 Yes
L009    M002    06/15/2023 Yes

  8 record(s) selected.

db2 => SELECT * FROM LOANDETAIL

LOANDETAIL_ID LOAN_ID BOOKDETAIL_ID RETURN_DATE
------------- ------- ------------- -----------
LD001         L002                1 -
LD002         L001                2 06/13/2023
LD003         L003                9 06/13/2023
LD004         L004                5 06/18/2023
LD005         L005                3 -
LD006         L006                1 -
LD007         L005                4 -
LD012         L009                1 06/19/2023
LD013         L0010               1 -

  9 record(s) selected.

db2 => SELECT * FROM FINE

FINE_ID LOANDETAIL_ID FINE_DATE  FINE_AMT PAYMENT_DATE PAYMENT_AMT
------- ------------- ---------- -------- ------------ -----------

  0 record(s) selected.

db2 =>
```

To invoke the stored procedure:

```
Call calculateFine()
```

To view the result:

```
Select * from fine
```



p/s: current date of screenshot is 19/6/2023 (fine_amt calculation will be affected by current date)

(d) Trigger to auto generate book id based on category keyword upon data insertion

```
CREATE TRIGGER generate_book_id
AFTER INSERT ON book
REFERENCING NEW AS new
FOR EACH ROW
BEGIN
  DECLARE new_cat_id VARCHAR(5);
  DECLARE book_count INT;

  SELECT cat_id INTO new_cat_id
  FROM category
  WHERE cat_keyword = (
    SELECT cat_keyword
    FROM category
    WHERE cat_id = new.cat_id
  );

  SELECT COUNT(*) INTO book_count
  FROM book
  WHERE cat_id = new_cat_id;

  UPDATE book
  SET book_id = (SELECT cat_keyword FROM category WHERE cat_id =
new_cat_id) || '' || VARCHAR(book_count),
      cat_id = new_cat_id
  WHERE book_id = new.book_id;
END
```

This trigger is to generate and assign a unique book_id for each newly inserted data in the book table. The new_cat_id will store the cat_id of the newly inserted data, while book_count will store the count of existing books for the same category. We first select the cat_id of the newly inserted data and fetch its cat_keyword, we then select the book_count of existing books with the same cat_id. After that, we update the book table by setting the **concatenation** of cat_keyword of the category and the book_count value. Book_count value is set to varchar since we need to have the same datatype for the concatenation.

To invoke the trigger:

```
db2 => insert into book values('','C002','B1',1,'Data Science for Beginners',2009)
```

Results:

```
db2 => insert into book values ('','C002','B1',1,'Data Science for Beginners',2009)
DB20000I  The SQL command completed successfully.
db2 => select * from book

BOOK_ID CAT_ID TAG_ID BOOK_AMT BOOK_TITLE                                          PUB_YEAR
------- ------ ------ -------- ---------------------------------------------------- -----------
CS1     C002   G1            3 Programming Fundamentals                                    2019
CS2     C002   G2            2 The Future of Computer-Assisted Education                   2015
CS3     C002   R1            1 Computer Architecture                                       2019
IT1     C001   B1            2 Textbook of Information Technology                          1988
LAW1    C003   Y1            3 Cambridge Law Journal                                       2021
MT1     C004   G1            3 Mathematic II - Trimester 1 2021/2022                       2021
CS4     C002   G1            1 Programming Fundamentals                                    2021
CS5     C002   B1            1 Data Science for Beginners                                  2009

  8 record(s) selected.
```

Contributions

1. Ang Jin Nan(1211100925) 100%
2. Chan Kar Kin(1211102630) 100%
3. Ng Yun Shi(1211103311) 100%
4. Tai Qi Tong(1211102777) 100%