University of Texas at Arlington

Project 2: Library Management System

Team Members:
Joshua Lian, Kierra Ashford, Abhinav Shrestha
Date:05/01/2023
CSE 3330-002

HONOR CODE

I pledge, on my honor, to uphold UT Arlington's tradition of academic integrity, a tradition that values hard work and honest effort in the pursuit of academic excellence. I promise that I will submit only work that I personally create or that I contribute to group collaborations, and I will appropriately reference any work from other sources. I will follow the highest standards of integrity and uphold the spirit of the Honor Code.

Abhinav Shrestha

Joshua Lian

Kierra Ashford

Task 1 Queries:

Query 1:

Alter table command: `ALTER TABLE BOOK_LOANS ADD Late BOOLEAN;`

Query:

```
UPDATE BOOK_LOANS SET Late = (CASE WHEN
JULIANDAY(returned_date) -JULIANDAY(due_date) <= 0 THEN 0
ELSE 1 END);
```

Result:

```
sqlite> SELECT * FROM BOOK_LOANS;
```

book_id	branch_id	card_no	date_out	due_date	returned_date	Late
1	1	123456	2022-01-01	2022-02-01	2022-02-01	0
2	1	789012	2022-01-02	2022-02-02	2022-02-02	0
3	2	345678	2022-01-03	2022-02-03	2022-02-03	0
4	3	901234	2022-01-04	2022-02-04	2022-02-04	0
5	1	567890	2022-01-05	2022-02-05	2022-02-09	1
6	2	234567	2022-01-06	2022-02-06	2022-02-10	1
7	2	890123	2022-01-07	2022-02-07	2022-03-08	1
8	3	456789	2022-01-08	2022-02-08	2022-03-10	1
9	1	111111	2022-01-09	2022-02-09	2022-02-06	0
10	2	222222	2022-01-10	2022-02-10	2022-02-07	0
11	1	333333	2022-03-01	2022-03-08	2022-02-08	0
12	3	444444	2022-03-03	2022-03-10	2022-03-10	0
13	3	555555	2022-02-03	2022-03-03	2022-02-18	0
14	1	565656	2022-01-14	2022-02-14	2022-03-31	1
15	3	676767	2022-01-15	2022-02-15	2022-02-21	1
16	2	787878	2022-03-05	2022-03-12	2022-02-24	0
17	3	989898	2022-03-23	2022-03-30	2022-03-30	0
18	3	121212	2022-01-18	2022-02-18	2022-02-18	0
19	1	232323	2022-03-24	2022-03-31	2022-03-31	0
20	3	343434	2022-01-21	2022-02-21	2022-02-21	0
21	3	454545	2022-01-24	2022-02-24	2022-02-24	0

The query added a Late column and set the values to 0 if not late and 1 if the book was returned late for all 21 BOOK_LOANS.

Query 2:

Alter table command: `ALTER TABLE LIBRARY_BRANCH ADD LateFee DOUBLE;`

Query:

```
UPDATE LIBRARY_BRANCH SET LateFee = 0.50 WHERE branch_id = 1;
UPDATE LIBRARY_BRANCH SET LateFee = 0.25 WHERE branch_id = 2;
UPDATE LIBRARY_BRANCH SET LateFee = 0.75 WHERE branch_id = 3;
```

Result:

```
sqlite> SELECT * FROM LIBRARY_BRANCH;
branch_id  branch_name   branch_address                  LateFee
---------  ------------  ------------------------------  -------
1          Main Branch   123 Main St, New York, NY 10003   0.5
2          West Branch   456 West St, Arizona, AR 70622    0.25
3          East Branch   789 East St, New Jersy, NY 32032  0.75
4          North Branch  456 NW,Irving,TX 76100           1.0
5          UTA Branch    123 Cooper St,Arlington TX 76101  1.25
```

The queries added a LateFee column and updated the values to the late fees according to the branch.

Query for VIEW:

```
CREATE VIEW vBookLoanInfo
AS SELECT B.card_no,B.name as Borrower_Name,
date_out,due_date,returned_date,
(JULIANDAY(returned_date)-JULIANDAY(date_out)) AS
TotalDays,title,CASE WHEN returned_date <= due_date THEN 0 ELSE
(JULIANDAY(returned_date)-JULIANDAY(due_date)) END AS
Days_late,LB.branch_id, CASE WHEN returned_date <= due_date THEN
0 ELSE ((JULIANDAY(returned_date)-JULIANDAY(due_date))*LateFee)
END AS Total_Late_Fee_Balance
FROM BOOK_LOANS AS BL, BORROWERS AS B, BOOK as BO,
LIBRARY_BRANCH AS LB
WHERE BL.book_id = BO.book_id AND B.card_no = BL.card_no AND
BL.branch_id = LB.branch_id
ORDER BY B.card_no ASC;
```

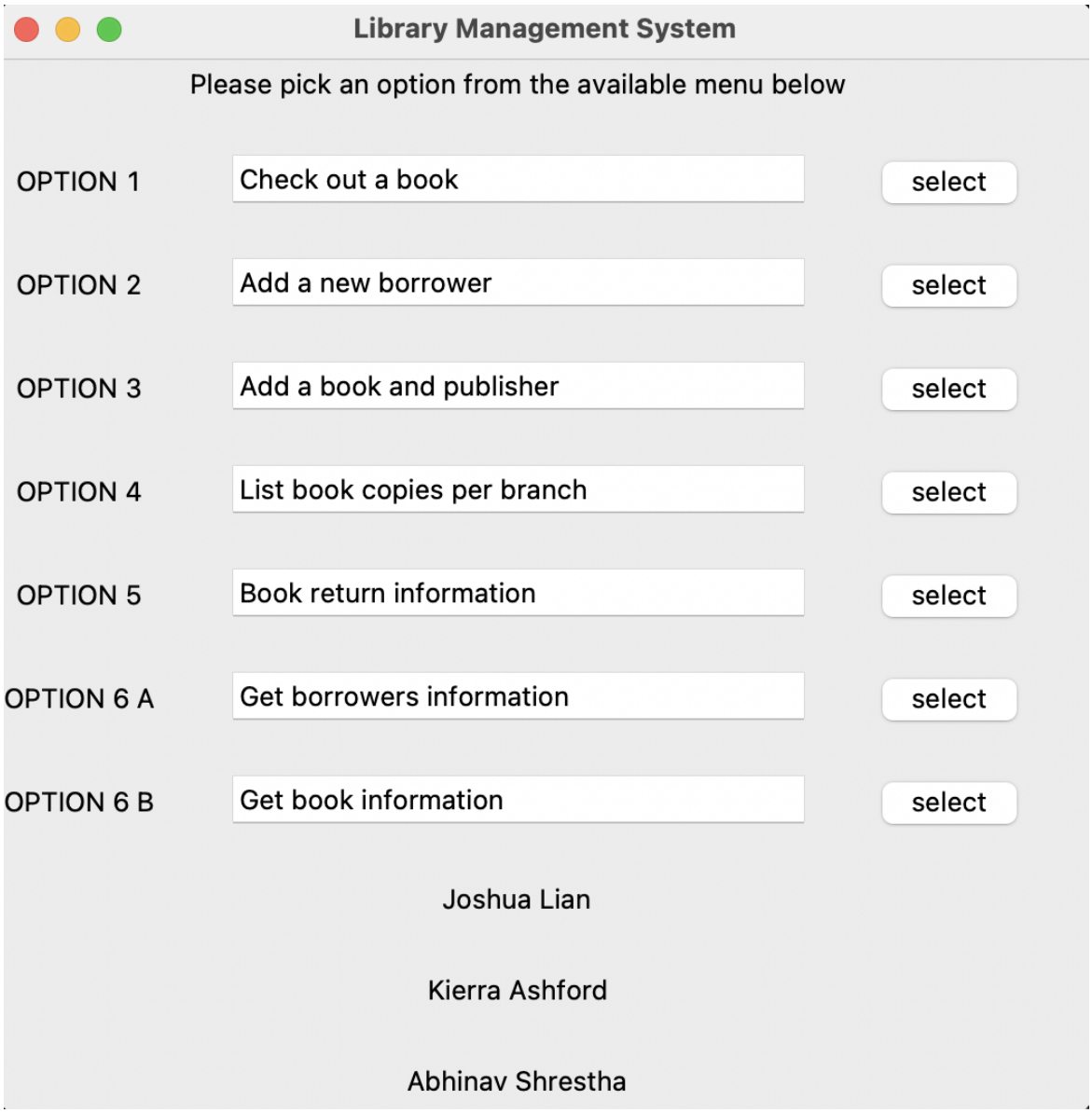
Screenshot of the Select view command output:

```
sqlite> select * from vBookLoanInfo;
card_no  Borrower_Name    date_out    due_date    returned_date  TotalDays  title                                Days_late  branch_id  Total_Late_Fee_Balance
-------  ---------------  ----------  ----------  -------------  ---------  -----------------------------------  ---------  ---------  ---------------------
111111   Alex Kim         2022-01-09  2022-02-09  2022-02-06     28.0       Brave New World                      0          1          0
121212   Chloe Park       2022-01-18  2022-02-18  2022-02-18     31.0       The Da Vinci Code                    0          3          0
123456   John Smith       2022-01-01  2022-02-01  2022-02-01     31.0       To Kill a Mockingbird                0          1          0
222222   Rachel Lee       2022-01-10  2022-02-10  2022-02-07     28.0       The Picture of Dorian Gray           0          2          0
232323   William Chen     2022-03-24  2022-03-31  2022-03-31     7.0        The Adventures of Huckleberry Finn   0          1          0
234567   Emily Lee        2022-01-06  2022-02-06  2022-02-10     35.0       Animal Farm                          4.0        2          1.0
333333   William Johnson  2022-03-01  2022-03-08  2022-02-08     -21.0      The Alchemist                        0          1          0
343434   Olivia Johnson   2022-01-21  2022-02-21  2022-02-21     31.0       The Adventures of Tom Sawyer         0          3          0
345678   Bob Johnson      2022-01-03  2022-02-03  2022-02-03     31.0       Pride and Prejudice                  0          2          0
444444   Ethan Martinez   2022-03-03  2022-03-10  2022-03-10     7.0        The God of Small Things              0          3          0
454545   Dylan Kim        2022-01-24  2022-02-24  2022-02-24     31.0       A Tale of Two Cities                 0          3          0
456789   Laura Chen       2022-01-08  2022-02-08  2022-03-10     61.0       Lord of the Flies                    30.0       3          22.5
555555   Grace Hernandez  2022-02-03  2022-03-03  2022-02-18     15.0       Wuthering Heights                    0          3          0
565656   Sophia Park      2022-01-14  2022-02-14  2022-03-31     76.0       The Hobbit                           45.0       1          22.5
567890   Tom Lee          2022-01-05  2022-02-05  2022-02-09     35.0       One Hundred Years of Solitude        4.0        1          2.0
676767   Olivia Lee       2022-01-15  2022-02-15  2022-02-21     37.0       The Lord of the Rings                6.0        3          4.5
787878   Noah Thompson    2022-03-05  2022-03-12  2022-02-24     -9.0       The Hitchhiker's Guide to the Galaxy 0          2          0
789012   Jane Doe         2022-01-02  2022-02-02  2022-02-02     31.0       1984                                 0          1          0
890123   Michael Park     2022-01-07  2022-02-07  2022-03-08     60.0       The Catcher in the Rye               29.0       2          7.25
901234   Sarah Kim        2022-01-04  2022-02-04  2022-02-04     31.0       The Great Gatsby                     0          3          0
989898   Olivia Smith     2022-03-23  2022-03-30  2022-03-30     7.0        The Diary of a Young Girl            0          3          0
sqlite> ›
```

Action output response: There were 21 rows returned by the select view command.

Task 2:

Parent GUI:



Description: The parent GUI is the main User Interface which is displayed when the code is run. There are 7 options for the user corresponding to the 7 different tasks which the GUI can perform. The text box has a short description of what the 'select' button is for. Upon pressing any of the 'select' buttons, a child GUI window opens up which asks for different inputs based on the functionality of the child GUI. The description of each child GUI along with the screenshots of the child GUI's are shown below.

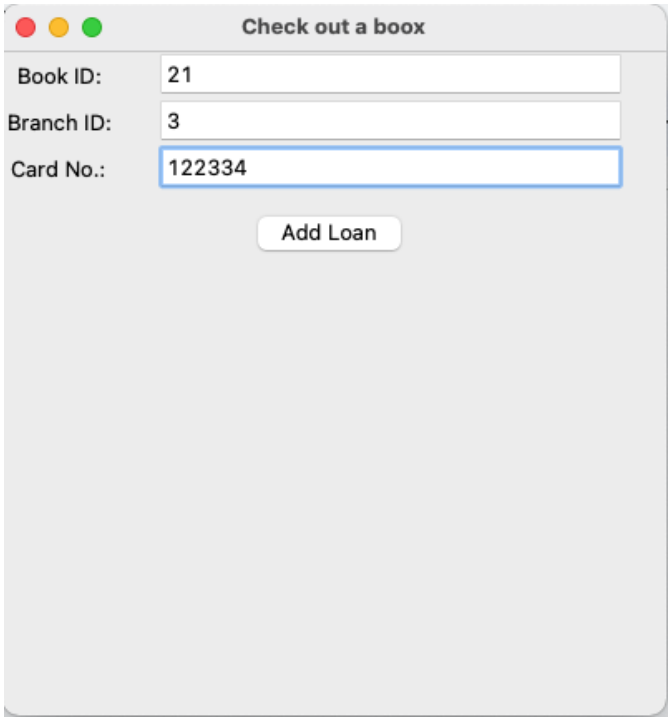
Requirement 1:

TRIGGER for Updating the BOOK_COPIES:

```
CREATE TRIGGER DEC_BOOK_COPIES
AFTER INSERT ON BOOK_LOANS
FOR EACH ROW
BEGIN
UPDATE BOOK_COPIES
SET num_of_copies = num_of_copies - 1
WHERE book_id = NEW.book_id AND branch_id = NEW.branch_id;
END;
```

Executable Query: check_cur.execute("INSERT INTO BOOK_LOANS (book_id, branch_id, card_no, date_out, due_date) VALUES (?,?,?, DATE('now'), DATE('now', '+30 days'))", (book_id, branch_id, card_no))

GUI View:



Description

The GUI takes input book_id, branch_id, and card_no of the borrower. The date_out is generated by the system itself using the DATE('now') function and assigns a due date which is the loaned date plus 30 days. Returned date and Late columns are assumed to be null because the book was just borrowed recently. The num_of_copies are also changed by the trigger while inserting values to BOOK_LOANS. The updates in the database are shown below.

Database:

BOOK_COPIES:

Before:

```
sqlite> SELECT * FROM BOOK_COPIES;
book_id  branch_id  num_of_copies
-------  ---------  -------------
1        1          3
2        1          2
3        2          1
4        3          4
5        1          5
6        2          3
7        2          2
8        3          1
9        1          4
10       2          2
11       1          3
12       3          2
13       3          1
14       1          5
15       3          1
16       2          3
17       3          2
18       3          2
19       1          5
20       3          1
21       3          1
```

After:

```
sqlite> SELECT * FROM BOOK_COPIES;
book_id  branch_id  num_of_copies
-------  ---------  -------------
1        1          3
2        1          2
3        2          1
4        3          4
5        1          5
6        2          3
7        2          2
8        3          1
9        1          4
10       2          2
11       1          3
12       3          2
13       3          1
14       1          5
15       3          1
16       2          3
17       3          2
18       3          2
19       1          5
20       3          1
21       3          0
```

BOOK_LOANS:

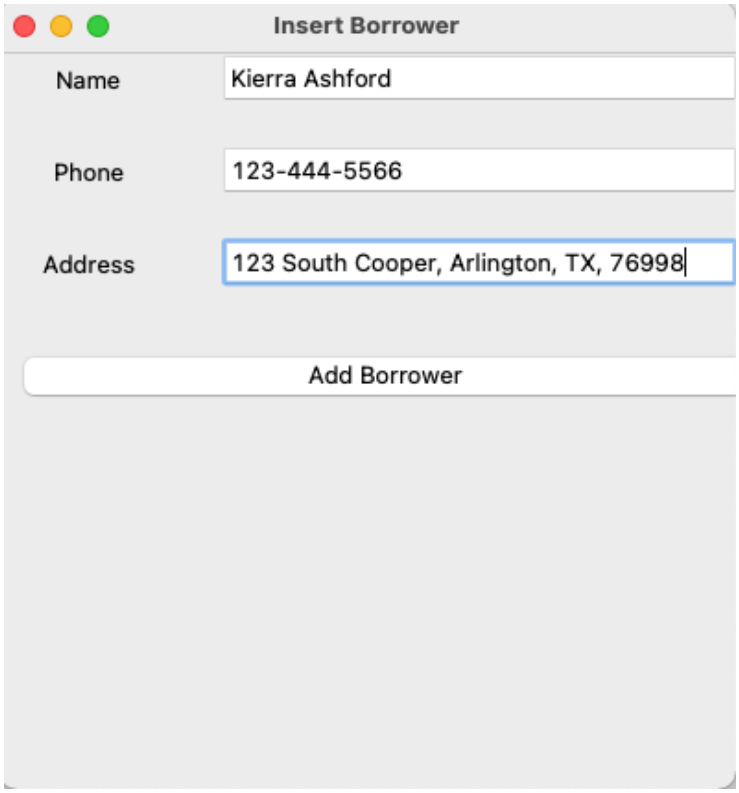
```
sqlite> SELECT * FROM BOOK_LOANS;
book_id  branch_id  card_no  date_out    due_date    returned_date  Late
-------  ---------  -------  ----------  ----------  -------------  ----
1        1          123456   2022-01-01  2022-02-01  2022-02-01     0
2        1          789012   2022-01-02  2022-02-02  2022-02-02     0
3        2          345678   2022-01-03  2022-02-03  2022-02-03     0
4        3          901234   2022-01-04  2022-02-04  2022-02-04     0
5        1          567890   2022-01-05  2022-02-05  2022-02-09     1
6        2          234567   2022-01-06  2022-02-06  2022-02-10     1
7        2          890123   2022-01-07  2022-02-07  2022-03-08     1
8        3          456789   2022-01-08  2022-02-08  2022-03-10     1
9        1          111111   2022-01-09  2022-02-09  2022-02-06     0
10       2          222222   2022-01-10  2022-02-10  2022-02-07     0
11       1          333333   2022-03-01  2022-03-08  2022-02-08     0
12       3          444444   2022-03-03  2022-03-10  2022-03-10     0
13       3          555555   2022-02-03  2022-03-03  2022-02-18     0
14       1          565656   2022-01-14  2022-02-14  2022-03-31     1
15       3          676767   2022-01-15  2022-02-15  2022-02-21     1
16       2          787878   2022-03-05  2022-03-12  2022-02-24     0
17       3          989898   2022-03-23  2022-03-30  2022-03-30     0
18       3          121212   2022-01-18  2022-02-18  2022-02-18     0
19       1          232323   2022-03-24  2022-03-31  2022-03-31     0
20       3          343434   2022-01-21  2022-02-21  2022-02-21     0
21       3          454545   2022-01-24  2022-02-24  2022-02-24     0
21       3          122334   2023-04-30  2023-05-30
```

Requirement 2:

Executable Query: submit_cur.execute("INSERT INTO BORROWERS (name,phone,address) VALUES (:name, :phone, :address) ",

```
{
        'name': name.get(),
        'phone': phone.get(),
        'address': address.get(),
})
```

GUI Implementation:



Description: The GUI takes input of the borrower name, phone, and the address. Upon pressing the Add Borrower button, the borrower gets added to the database and also gets assigned with a card_no. The database table information is shown below.

Database:

Before:

```
sqlite> SELECT * FROM BORROWERS;
card_no  name             address                                  phone
-------  ---------------  ---------------------------------------  ------------
111111   Alex Kim         983 Sine St, Arizona, AR 70451           678-784-5563
121212   Chloe Park       345 Shark St, Arizona, AR 72213          755-905-5572
123456   John Smith       456 Oak St, Arizona, AR 70010            205-555-5555
222222   Rachel Lee       999 Apple Ave, Arizona, AR 70671         231-875-5564
232323   William Chen     890 Sting St, New York, NY 10459         406-755-5580
234567   Emily Lee        389 Oaklay St, Arizona, AR 70986         231-678-5560
333333   William Johnson  705 Paster St, New Jersey 32002         235-525-5567
343434   Olivia Johnson   345 Pine St, New Jersey, NJ 32095        662-554-5575
345678   Bob Johnson      12 Elm St, Arizona, AR 70345             545-234-5557
444444   Ethan Martinez   466 Deeplm St, New York, NY 10321        555-555-5569
454545   Dylan Kim        567 Cowboy way St, New Jersey, NJ 32984  435-254-5578
456789   Laura Chen       345 Mapman Ave, Arizona, AR 70776        565-985-9962
555555   Grace Hernandez  315 Babes St, Arizona, AR 70862          455-567-5587
565656   Sophia Park      678 Dolphin St, New York, NY 10062       675-455-5568
567890   Tom Lee          678  S Oak St, New York, NY 10045        209-525-5559
676767   Olivia Lee       345 Spine St, New York, NY 10092         435-878-5569
787878   Noah Thompson    189 GreenOak Ave, New Jersey, NJ 32453   245-555-5571
789012   Jane Doe         789 Maple Ave, New Jersey, NJ 32542      555-235-5556
890123   Michael Park     123 Pinewood St, New Jersey, NJ 32954    655-890-2161
901234   Sarah Kim        345 Pine St, New York, NY 10065          515-325-2158
989898   Olivia Smith     178 Elm St, New Jersey, NJ 32124         325-500-5579
```

After:

```
sqlite> SELECT * FROM BORROWERS;
card_no  name             address                                  phone
-------  ---------------  ---------------------------------------  ------------
111111   Alex Kim         983 Sine St, Arizona, AR 70451           678-784-5563
121212   Chloe Park       345 Shark St, Arizona, AR 72213          755-905-5572
123456   John Smith       456 Oak St, Arizona, AR 70010            205-555-5555
222222   Rachel Lee       999 Apple Ave, Arizona, AR 70671         231-875-5564
232323   William Chen     890 Sting St, New York, NY 10459         406-755-5580
234567   Emily Lee        389 Oaklay St, Arizona, AR 70986         231-678-5560
333333   William Johnson  705 Paster St, New Jersey 32002         235-525-5567
343434   Olivia Johnson   345 Pine St, New Jersey, NJ 32095        662-554-5575
345678   Bob Johnson      12 Elm St, Arizona, AR 70345             545-234-5557
444444   Ethan Martinez   466 Deeplm St, New York, NY 10321        555-555-5569
454545   Dylan Kim        567 Cowboy way St, New Jersey, NJ 32984  435-254-5578
456789   Laura Chen       345 Mapman Ave, Arizona, AR 70776        565-985-9962
555555   Grace Hernandez  315 Babes St, Arizona, AR 70862          455-567-5587
565656   Sophia Park      678 Dolphin St, New York, NY 10062       675-455-5568
567890   Tom Lee          678  S Oak St, New York, NY 10045        209-525-5559
676767   Olivia Lee       345 Spine St, New York, NY 10092         435-878-5569
787878   Noah Thompson    189 GreenOak Ave, New Jersey, NJ 32453   245-555-5571
789012   Jane Doe         789 Maple Ave, New Jersey, NJ 32542      555-235-5556
890123   Michael Park     123 Pinewood St, New Jersey, NJ 32954    655-890-2161
901234   Sarah Kim        345 Pine St, New York, NY 10065          515-325-2158
989898   Olivia Smith     178 Elm St, New Jersey, NJ 32124         325-500-5579
989899   Kierra Ashford   123 South Cooper, Arlington, TX, 76998   123-444-5566
```

Requirement 3:

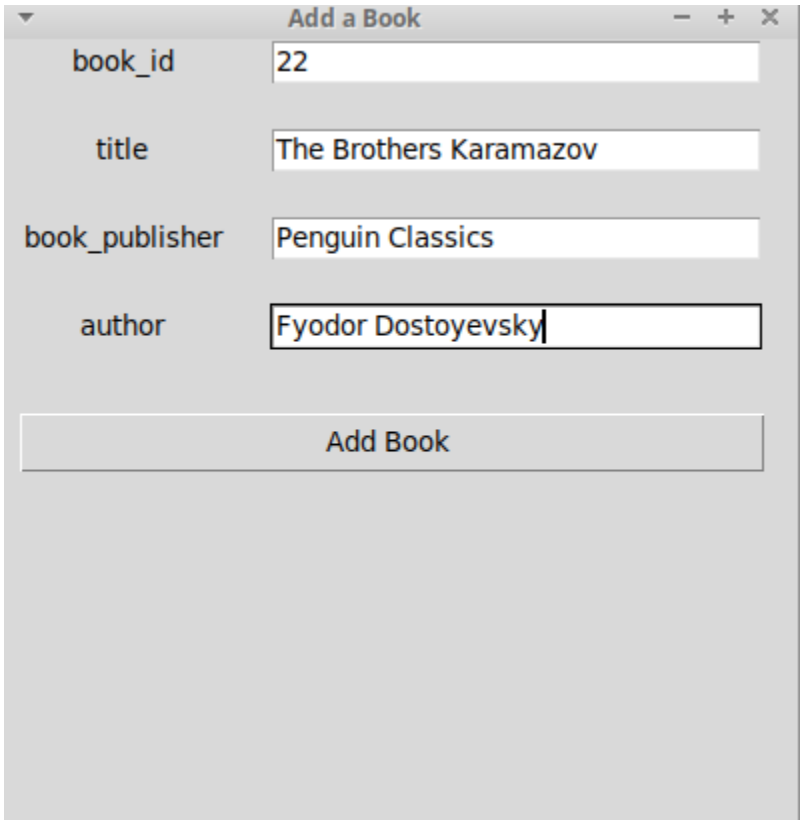
Executable Query: option_three_cur.execute("INSERT INTO BOOK VALUES (:book_id, :title, :book_publisher) ",

```
{   'book_id': book_id.get(),
        'title': title.get(),
        'book_publisher': book_publisher.get(),



})
option_three_cur.execute("INSERT INTO BOOK_AUTHORS VALUES (:book_id, :author)",
                {
                        'book_id': book_id.get(),
                        'author' : author.get(),
                })
```

GUI Implementation:



BEFORE:

```
sqlite> SELECT * FROM (BOOK NATURAL JOIN BOOK_COPIES) NATURAL JOIN BOOK_AUTHORS;
book_id  title                              book_publisher              branch_id  num_of_copies  author_name
-------  ---------------------------------  --------------------------  ---------  -------------  --------------------
1        To Kill a Mockingbird              HarperCollins               1          3              Harper Lee
2        1984                               Penguin Books               1          2              George Orwell
3        Pride and Prejudice                Penguin Classics            2          1              Jane Austen
4        The Great Gatsby                   Scribner                    3          4              F. Scott Fitzgerald
5        One Hundred Years of Solitude      Harper & Row                1          5              Gabriel Garcia Marquez
6        Animal Farm                        Penguin Books               2          3              George Orwell
7        The Catcher in the Rye             Little, Brown and Company   2          2              J.D. Salinger
8        Lord of the Flies                  Faber and Faber             3          1              William Golding
9        Brave New World                    Chatto & Windus             1          4              Aldous Huxley
10       The Picture of Dorian Gray         Ward, Lock and Co.          2          2              Oscar Wilde
11       The Alchemist                      HarperCollins               1          3              Paulo Coelho
12       The God of Small Things            Random House India          3          2              Arundhati Roy
13       Wuthering Heights                  Thomas Cautley Newby        3          1              Emily Bronte
14       The Hobbit                         Allen & Unwin               1          5              J.R.R. Tolkien
15       The Lord of the Rings              Allen & Unwin               3          1              J.R.R. Tolkien
16       The Hitchhiker's Guide to the Galaxy  Pan Books                2          3              Douglas Adams
17       The Diary of a Young Girl          Bantam Books                3          2              Anne Frank
18       The Da Vinci Code                  Doubleday                   3          2              Dan Brown
19       The Adventures of Huckleberry Finn  Penguin Classics           1          5              Mark Twain
20       The Adventures of Tom Sawyer       American Publishing Company  3         1              Mark Twain
21       A Tale of Two Cities               Chapman and Hall            3          1              Charles Dickens
```

AFTER:

```
sqlite> SELECT * FROM (BOOK NATURAL JOIN BOOK_COPIES) NATURAL JOIN BOOK_AUTHORS;
book_id  title                              book_publisher              branch_id  num_of_copies  author_name
-------  ---------------------------------  --------------------------  ---------  -------------  --------------------
1        To Kill a Mockingbird              HarperCollins               1          3              Harper Lee
2        1984                               Penguin Books               1          2              George Orwell
3        Pride and Prejudice                Penguin Classics            2          1              Jane Austen
4        The Great Gatsby                   Scribner                    3          4              F. Scott Fitzgerald
5        One Hundred Years of Solitude      Harper & Row                1          5              Gabriel Garcia Marquez
6        Animal Farm                        Penguin Books               2          3              George Orwell
7        The Catcher in the Rye             Little, Brown and Company   2          2              J.D. Salinger
8        Lord of the Flies                  Faber and Faber             3          1              William Golding
9        Brave New World                    Chatto & Windus             1          4              Aldous Huxley
10       The Picture of Dorian Gray         Ward, Lock and Co.          2          2              Oscar Wilde
11       The Alchemist                      HarperCollins               1          3              Paulo Coelho
12       The God of Small Things            Random House India          3          2              Arundhati Roy
13       Wuthering Heights                  Thomas Cautley Newby        3          1              Emily Bronte
14       The Hobbit                         Allen & Unwin               1          5              J.R.R. Tolkien
15       The Lord of the Rings              Allen & Unwin               3          1              J.R.R. Tolkien
16       The Hitchhiker's Guide to the Galaxy  Pan Books                2          3              Douglas Adams
17       The Diary of a Young Girl          Bantam Books                3          2              Anne Frank
18       The Da Vinci Code                  Doubleday                   3          2              Dan Brown
19       The Adventures of Huckleberry Finn  Penguin Classics           1          5              Mark Twain
20       The Adventures of Tom Sawyer       American Publishing Company  3         1              Mark Twain
21       A Tale of Two Cities               Chapman and Hall            3          1              Charles Dickens
22       The Brothers Karamazov             Penguin Classics            1          5              Fyodor Dostoyevsky
22       The Brothers Karamazov             Penguin Classics            2          5              Fyodor Dostoyevsky
22       The Brothers Karamazov             Penguin Classics            3          5              Fyodor Dostoyevsky
22       The Brothers Karamazov             Penguin Classics            4          5              Fyodor Dostoyevsky
22       The Brothers Karamazov             Penguin Classics            5          5              Fyodor Dostoyevsky
```

TRIGGER STATEMENT:

CREATE TRIGGER BookUpdate

After Insert

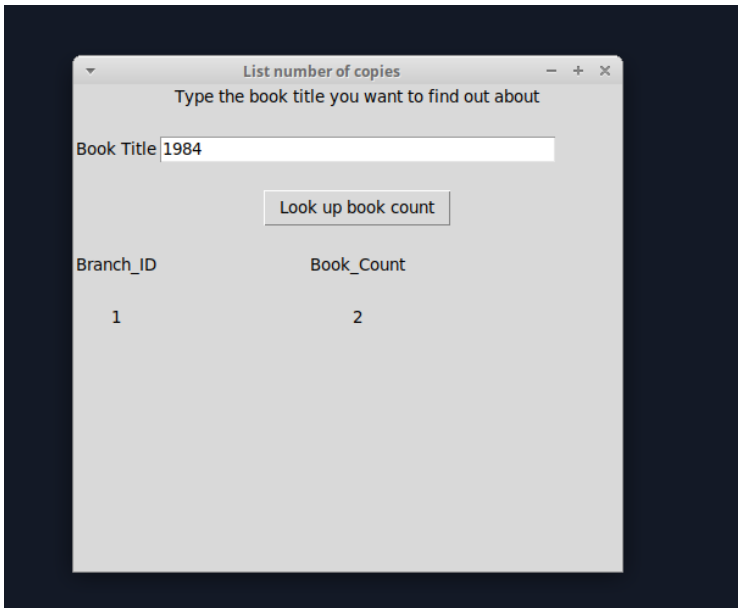
On Book
BEGIN
INSERT INTO Book_Copies VALUES (new.book_id, 1, 5);
INSERT INTO Book_Copies VALUES (new.book_id, 2, 5);
INSERT INTO Book_Copies VALUES (new.book_id, 3, 5);
INSERT INTO Book_Copies VALUES (new.book_id, 4, 5) ;
INSERT INTO Book_Copies VALUES (new.book_id, 5, 5);

END;

- Assumptions made for this query: We had to change the primary key constraint of book_id for Book_Copies to foreign key because the instructions required that for a book insert we need to add copies of that book to each of the 5 branches, with 5 copies at each branch.

Requirement 4:

An output for the book 1984



Query : SELECT branch_id, num_of_copies FROM BOOK_COPIES NATURAL JOIN BOOK WHERE title = ? GROUP BY branch_id",(book.get(),)

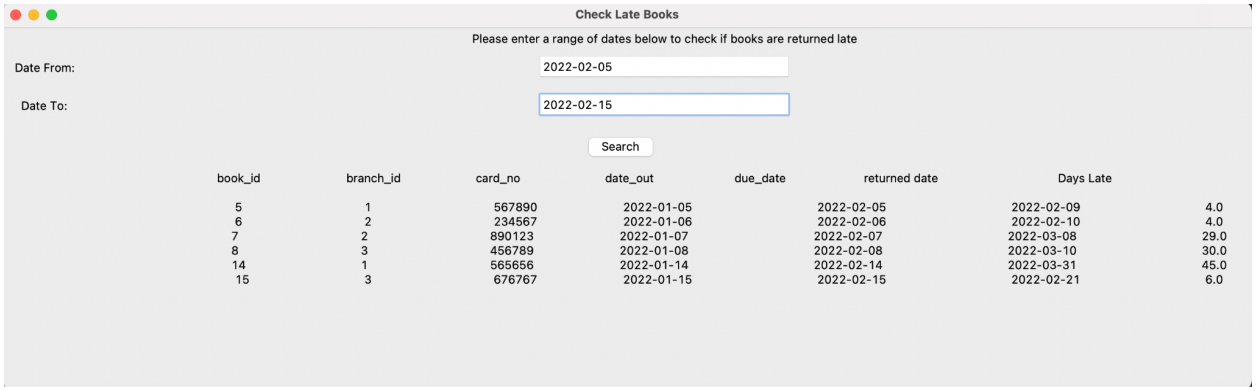
Description:

The purpose of this task is to print out the number of copies that are in each branch. As of now, we have made our book_id a primary key. For that reason, a certain book is only located in one library. Because of that, it never has more then one column of data to return. If there was more than one library that the book was located in, then it would return the branches and list the count for each branch. The user would type in the book tile in the “Book Title” section, if the book is not typed correctly, then it will only output the attributes and nothing more.

Requirement 5:

Executable Query: search_cur.execute("SELECT book_id, branch_id, card_no, date_out, due_date, returned_date, (JULIANDAY(returned_date)- JULIANDAY(due_date)) AS Days_late FROM BOOK_LOANS WHERE returned_date > due_date AND (due_date > ? AND due_date < ?)", (date_from, date_to))

GUI View:



Description:

For this GUI, it takes two inputs, one if the date from and the other is date to from the due date. This GUI will display the books that were returned late from the date from to the date to including the entered dates. It will display the book_id, branch_id, card_no, date_out, due_date, returned_date and the number of days late as Days Late. The padding for this GUI did not work properly even after adjusting them for several hours. The above result is the best we could get out of it.

Requirement 6:

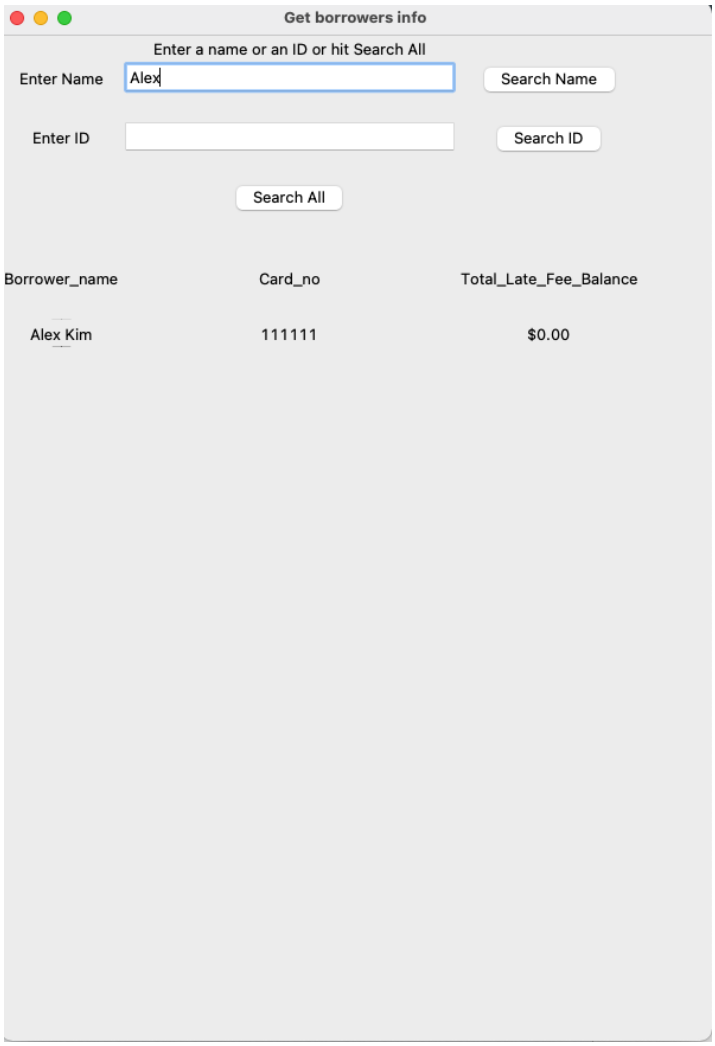
Part 6A:

For this GUI, it takes two inputs, the name and the card_no of the borrower. However, the user does not have to provide both the inputs. Users can search by either just the name entered and press the Search Name button or enter jus the card_no and press the Search ID button. Additionally, the GUI also has another button called Search All, which can be pressed without entering any inputs. The result of Search all shows all the borrowers ordered by the total late fee balance due.The sub parts along with their description are as follows:

Executable Query(Search by Name or part of the name):

submit_cur.execute("select Borrower_name, card_no, Total_Late_Fee_Balance FROM vBookLoanInfo WHERE Borrower_name LIKE '%'||:name||'%'",(name.get(),))

GUI Implementation:



Description: User can enter the full name or just part of the name in the Enter Name text box. On pressing the Search Name button, the GUI displays the Borrower_name, Card_no, and the total late fee balance from all the records matching the name or the part of the name.

Executable Query (Search by Borrower ID):

```
submit_cur.execute("select Borrower_name, card_no, Total_Late_Fee_Balance from
vBookLoanInfo where card_no = :card_no" ,
                    {
                        'card_no':ID.get(),
                    })
```

GUI Implementation:

Description: User can also search using the card_no of the borrower using the Enter ID text box. On pressing the Search ID button, the GUI displays the Borrower_name, card_no, and the total late fee balance for the matched Borrower ID.

Executable Query (Search with no filters or criteria):

submit_cur.execute("SELECT Borrower_name, card_no, Total_late_Fee_Balance FROM vBookLoanInfo ORDER BY Total_Late_Fee_Balance")

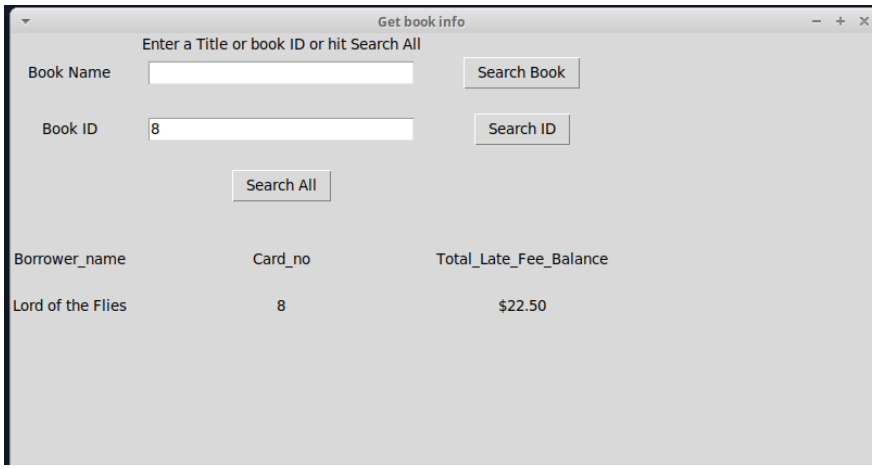
GUI Implementation:

Borrower_name	Card_no	Total_Late_Fee_Balance
John Smith	123456	$0.00
Jane Doe	789012	$0.00
Bob Johnson	345678	$0.00
Sarah Kim	901234	$0.00
Alex Kim	111111	$0.00
Rachel Lee	222222	$0.00
William Johnson	333333	$0.00
Ethan Martinez	444444	$0.00
Grace Hernandez	555555	$0.00
Noah Thompson	787878	$0.00
Olivia Smith	989898	$0.00
Chloe Park	121212	$0.00
William Chen	232323	$0.00
Olivia Johnson	343434	$0.00
Dylan Kim	454545	$0.00
Emily Lee	234567	$1.00
Tom Lee	567890	$2.00
Olivia Lee	676767	$4.50
Michael Park	890123	$7.25
Laura Chen	456789	$22.50
Sophia Park	565656	$22.50

Description: Users can simply press the Search ALL button without entering anything in the text boxes to display all the borrowers with their card_no and the remaining late fee balance. The GUI orders the list according to the total late fee balance.

Part 6B:

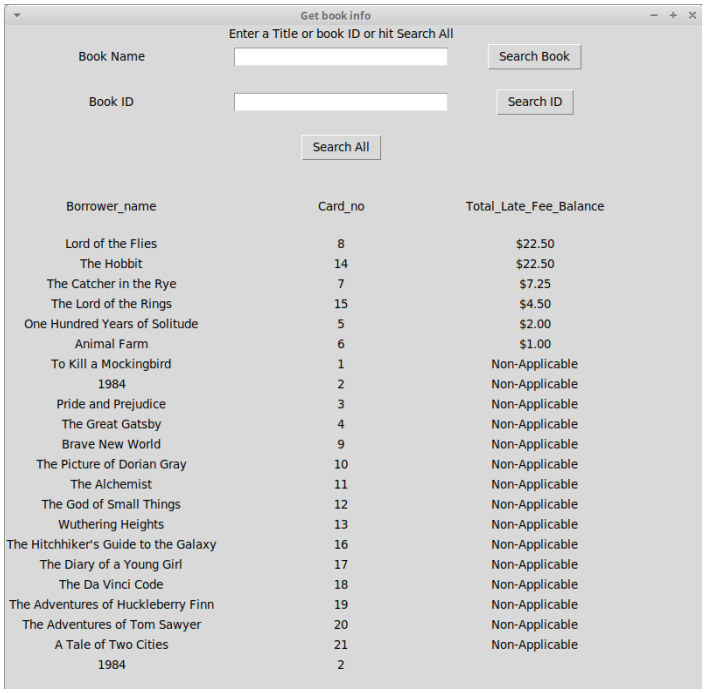
Outputs for the three sections of the GUI:



Book_ID ^

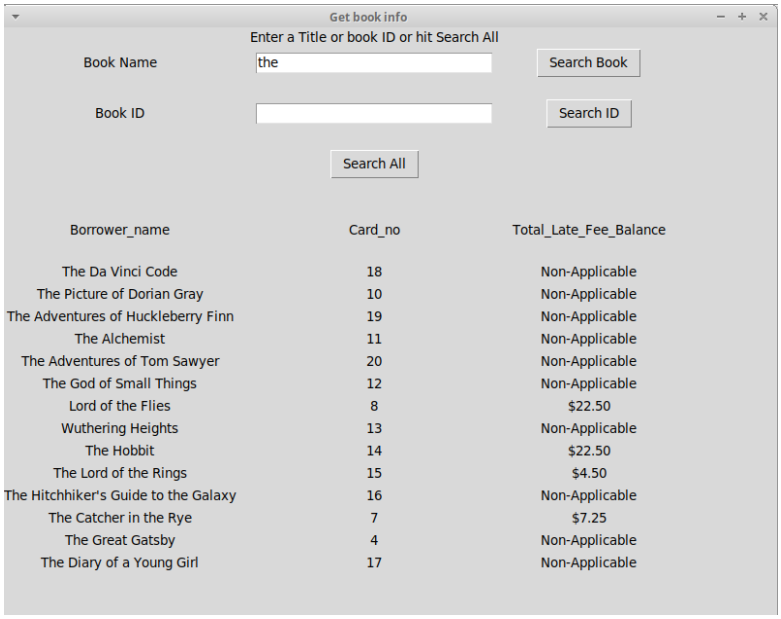
Query: SELECT BV.title, B.book_id, Total_late_Fee_Balance FROM (vBookLoanInfo AS BV JOIN BOOK AS B on B.title = BV.title) WHERE book_id = :book_id" ,

```
{
        'book_id':B_ID.get(),
})
```



Search All^

Query: SELECT BV.title, B.book_id, Total_late_Fee_Balance FROM (vBookLoanInfo AS BV JOIN BOOK AS B on B.title = BV.title) ORDER BY Total_Late_Fee_Balance DESC



Book Name^

Query: submit_cur.execute("SELECT BV.title, B.book_id, Total_late_Fee_Balance FROM (vBookLoanInfo AS BV JOIN BOOK AS B on B.title = BV.title) WHERE BV.title LIKE '%'||:name||'%'",(B_name.get(),))

Description:

For this task, the GUI is required to be able to have three functionalities, listing by book name, book ID, and nothing at all. This is very similar to the structure of 6.a. The logic used for these two are very identical. It starts off by giving you three options to choose from. If you press the select all button, then the output will be everything in the view ordered in ascending order of fee owed. The thing about this GUI is that you cannot continuously use the other function before exiting out of the one you are using. This means that you have to exit the GUI and press on it again from the parent GUI to get the three options again. You cannot try to type in book id after you have pressed select all or entered a name. Another thing is, when the output is unknown, then it will print out only the attributes in the column and nothing else. The parent GUI is the GUI that you first encounter. We based our front end that way. To switch GUIs for every available task. This allows us to have a cleaner look when we run out GUIs.

Labor Contribution

QUERY COMPLETION of TASK 1	
1	Joshua Lian
2	Kierra Ashford
3 (View)	Abhinav Shrestha

REQUIREMENT COMPLETION of TASK 2	
1,5,6A,6B	Abhinav Shrestha
4,6A, 6B	Joshua Lian
2,3.6A	Kierra Ashford