Project #2 Part 3 - CSE 3330-004 - Library Management System Database

Dev Patel

Task 1: Execute the following queries on the LMS database tables:

Query 1

Add an extra column ‘Late’ to the Book_Loan table. Values will be 0-for non-late returns, and 1-for late-returns. Then update the ‘Late’ column with '1' for all records that they have a return date later than the due date and with '0' for those were returned on time.

Query:

```
ALTER TABLE BOOK_LOANS ADD COLUMN Late INTEGER DEFAULT 0;
UPDATE BOOK_LOANS SET Late = CASE WHEN Returned_Date > Due_Date
THEN 1 ELSE 0 END;
```

Table:

```
CREATE TABLE Book_Loans (
    Book_ID INT,
    Branch_ID INT,
    Card_No INT,
    Date_Out DATE,
    Due_Date DATE,
    Returned_Date DATE,
    Late INTEGER DEFAULT 0,
    PRIMARY KEY (Book_ID, Branch_ID, Card_No),
    FOREIGN KEY (Book_ID) REFERENCES BOOK(Book_ID),
    FOREIGN KEY (Branch_ID) REFERENCES LIBRARY_BRANCH(Branch_ID),
    FOREIGN KEY (Card_No) REFERENCES Borrower(Card_No)
}
```

```
sqlite> ALTER TABLE BOOK_LOANS ADD COLUMN Late INTEGER DEFAULT 0;
sqlite> UPDATE BOOK_LOANS
   ...> SET Late = CASE
   ...>     WHEN Returned_Date > Due_Date THEN 1
   ...>     ELSE 0
   ...> END;
sqlite> SELECT * FROM BOOK_LOANS;
1|1|123456|2022-01-01|2022-02-01|2022-02-01|0
2|1|789012|2022-01-02|2022-02-02||0
3|2|345678|2022-01-03|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|2022-03-08|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|2022-03-24|1
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|2022-03-21|2022-02-21|0
21|3|454545|2022-01-24|2022-02-24|2022-02-24|0
sqlite>
```

Query 2

Add an extra column 'LateFee' to the Library_Branch table, decide late fee per day for each branch and update that column.

Query:

```
ALTER TABLE Library_Branch ADD COLUMN LateFee REAL;
UPDATE LIBRARY_BRANCH
SET LateFee = CASE
WHEN Branch_ID = 1 THEN 0.50
WHEN Branch_ID = 2 THEN 0.75
WHEN Branch_ID = 3 THEN 1.00
ELSE 0.60
END;
```

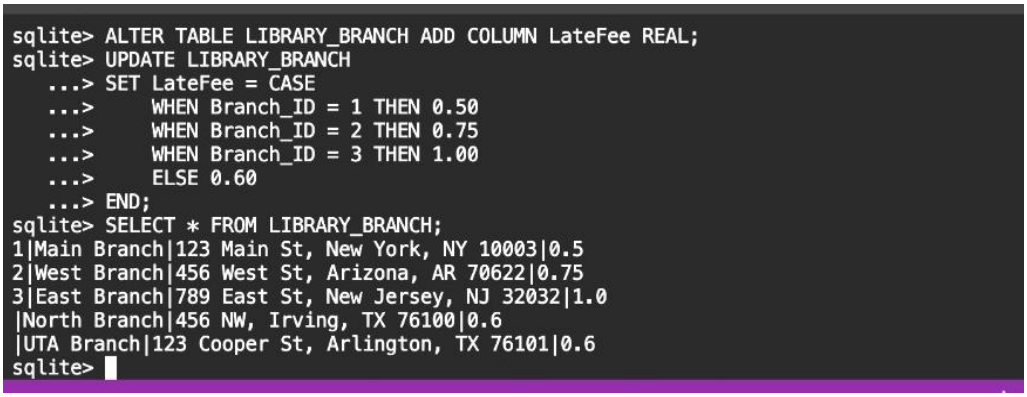
- *for Branch_ID 1 (Main Branch), made it a LateFee of 0.50*
- *for Branch_ID 2 (West Branch), made it a LateFee of 0.75*
- *for Branch_ID 3 (East Branch), made it a LateFee of 1.00*
- *For any other branches, set LateFee to 0.60*

Table:

```
CREATE TABLE LIBRARY_BRANCH (
Branch_ID INT PRIMARY KEY,
Branch_Name VARCHAR(200),
Branch_Address VARCHAR(200),
LateFee REAL);
```

Current Data in Table:

```
1|Main Branch|123 Main St, New York, NY 10003|0.5
2|West Branch|456 West St, Arizona, AR 70622|0.75
3|East Branch|789 East St, New Jersey, NJ 32032|1.0
|North Branch|456 NW, Irving, TX 76100|0.6
|UTA Branch|123 Cooper St, Arlington, TX 76101|0.6
```



```
sqlite> ALTER TABLE LIBRARY_BRANCH ADD COLUMN LateFee REAL;
sqlite> UPDATE LIBRARY_BRANCH
   ...> SET LateFee = CASE
   ...>     WHEN Branch_ID = 1 THEN 0.50
   ...>     WHEN Branch_ID = 2 THEN 0.75
   ...>     WHEN Branch_ID = 3 THEN 1.00
   ...>     ELSE 0.60
   ...> END;
sqlite> SELECT * FROM LIBRARY_BRANCH;
1|Main Branch|123 Main St, New York, NY 10003|0.5
2|West Branch|456 West St, Arizona, AR 70622|0.75
3|East Branch|789 East St, New Jersey, NJ 32032|1.0
|North Branch|456 NW, Irving, TX 76100|0.6
|UTA Branch|123 Cooper St, Arlington, TX 76101|0.6
sqlite>
```

Query 3

Create a **view** vBookLoanInfo that retrieves all information per book loan. The view should have the following attributes:

•Card_No,
•Borrower Name
•Date_Out,
•Due_Date,
•Returned_date,
•Total Days of book loaned out as *'TotalDays'* – *need to change weeks to days*
•Book Title,
•Number of days returned late *–if returned before or on due_date place zero*
•Branch ID,
•Total Late Fee Balance 'LateFeeBalance' *–If the book was not returned late than fee='0'*

Query:

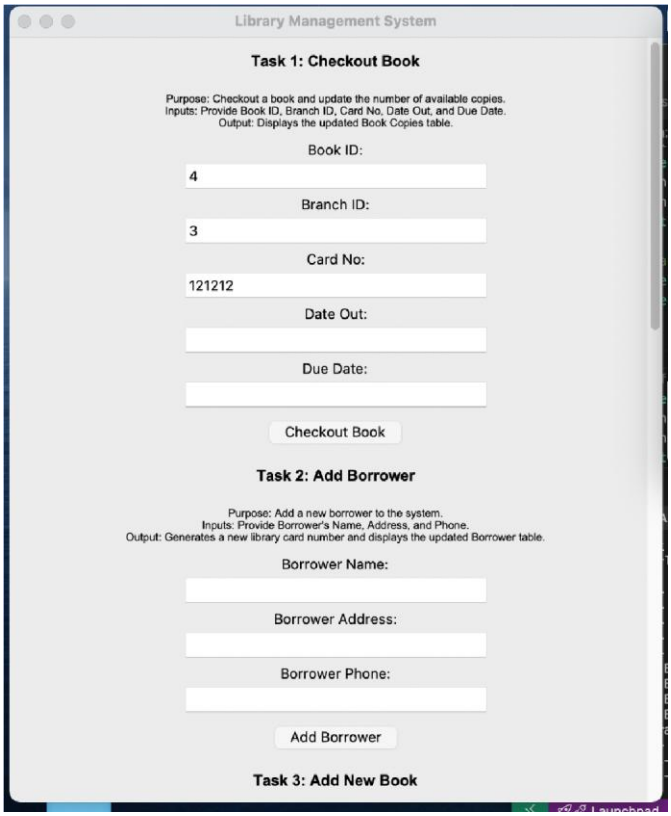
```
CREATE VIEW vBookLoanInfo AS
SELECT
    BOOK_LOANS.Card_No,
    BOOK.Title,
    BOOK_LOANS.Date_Out,
    BOOK_LOANS.Due_Date,
    BOOK_LOANS.Returned_Date,
    julianday(BOOK_LOANS.Returned_Date) -
julianday(BOOK_LOANS.Date_Out) AS TotalDays,
    BOOK.Title,
    CASE
        WHEN BOOK_LOANS.Late = 1 THEN
julianday(BOOK_LOANS.Returned_Date) - julianday(BOOK_LOANS.Due_Date)
ELSE 0
    END,
    BOOK_LOANS.Branch_ID,
    CASE
        WHEN BOOK_LOANS.Late = 1 THEN
            (julianday(BOOK_LOANS.Returned_Date) -
julianday(BOOK_LOANS.Due_Date)) * LIBRARY_BRANCH.LateFee
        ELSE 0
    END AS LateFeeBalance
FROM
    BOOK_LOANS
JOIN
    BORROWER ON BOOK_LOANS.Card_No = BORROWER.Card_No
JOIN
```

```
    BOOK ON BOOK_LOANS.Book_ID = BOOK.Book_ID
JOIN
    LIBRARY_BRANCH ON BOOK_LOANS.Branch_ID = LIBRARY_BRANCH.Branch_ID;
```

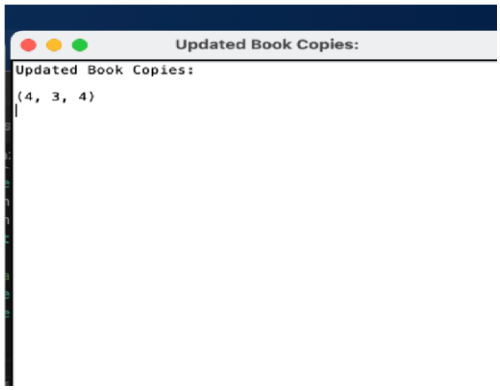
```
sqlite> CREATE VIEW vBookLoanInfo AS
   ...> SELECT
   ...>     BOOK_LOANS.Card_No,
   ...>     BORROWER.Name AS Borrower_Name,
   ...>     BOOK_LOANS.Date_Out,
   ...>     BOOK_LOANS.Due_Date,
   ...>     BOOK_LOANS.Returned_Date,
   ...>     JULIANDAY(BOOK_LOANS.Returned_Date) - JULIANDAY(BOOK_LOANS.Date_Out) AS TotalDays,
   ...>     BOOK.Title,
   ...>     CASE
   ...>         WHEN BOOK_LOANS.Late = 1 THEN
   ...>             JULIANDAY(BOOK_LOANS.Returned_Date) - JULIANDAY(BOOK_LOANS.Due_Date)
   ...>         ELSE 0
   ...>     END AS DaysLate,
   ...>     BOOK_LOANS.Branch_ID,
   ...>     CASE
   ...>         WHEN BOOK_LOANS.Late = 1 THEN
   ...>             (JULIANDAY(BOOK_LOANS.Returned_Date) - JULIANDAY(BOOK_LOANS.Due_Date)) * LIBRARY_BRANCH.LateFee
   ...>         ELSE 0
   ...>     END AS LateFeeBalance
   ...> FROM
   ...>     BOOK_LOANS
   ...> JOIN
   ...>     BORROWER ON BOOK_LOANS.Card_No = BORROWER.Card_No
   ...> JOIN
   ...>     BOOK ON BOOK_LOANS.Book_ID = BOOK.Book_ID
   ...> JOIN
   ...>     LIBRARY_BRANCH ON BOOK_LOANS.Branch_ID = LIBRARY_BRANCH.Branch_ID;
sqlite> SELECT * FROM vBookLoanInfo;
123456|John Smith|2022-01-01|2022-02-01|2022-02-01|31.0|To Kill a Mockingbird|0|1|0
789012|Jane Doe|2022-01-02|2022-02-02|||1984|0|1|0
345678|Bob Johnson|2022-01-03|2022-02-03|||Pride and Prejudice|0|2|0
901234|Sarah Kim|2022-01-04|2022-02-04|2022-02-04|31.0|The Great Gatsby|0|3|0
567890|Tom Lee|2022-01-05|2022-02-05|2022-02-09|35.0|One Hundred Years of Solitude|4.0|1|2.0
234567|Emily Lee|2022-01-06|2022-02-06|2022-02-10|35.0|Animal Farm|4.0|2|3.0
890123|Michael Park|2022-01-07|2022-02-07|2022-03-08|60.0|The Catcher in the Rye|29.0|2|21.75
456789|Laura Chen|2022-01-08|2022-02-08|2022-03-10|61.0|Lord of the Flies|30.0|3|30.0
111111|Alex Kim|2022-01-09|2022-02-09|2022-02-06|28.0|Brave New World|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
333333|William Johnson|2022-03-01|2022-03-08|2022-03-08|7.0|The Alchemist|0|1|0
444444|Ethan Martinez|2022-03-03|2022-03-10|2022-03-10|7.0|The God of Small Things|0|3|0
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
676767|Olivia Lee|2022-01-15|2022-02-15|2022-02-21|37.0|The Lord of the Rings|6.0|3|6.0
787878|Noah Thompson|2022-03-05|2022-03-12|2022-03-24|19.0|The Hitchhiker's Guide to the Galaxy|12.0|2|9.0
989898|Olivia Smith|2022-03-23|2022-03-30|2022-03-30|7.0|The Diary of a Young Girl|0|3|0
121212|Dylan Kim|2022-01-18|2022-02-18|2022-02-18|31.0|The Da Vinci Code|0|3|0
232323|William Chen|2022-03-24|2022-03-31|2022-03-31|7.0|The Adventures of Huckleberry Finn|0|1|0
343434|Olivia Johnson|2022-01-21|2022-03-21|2022-02-21|31.0|The Adventures of Tom Sawyer|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
sqlite>
```

Task 2: Create a GUI for the LMS database:

1. User checks out a book, add it to Book_Loan, the number of copies needs to be updated via trigger in the Book_Copies table. Show the output of the updated Book_Copies.



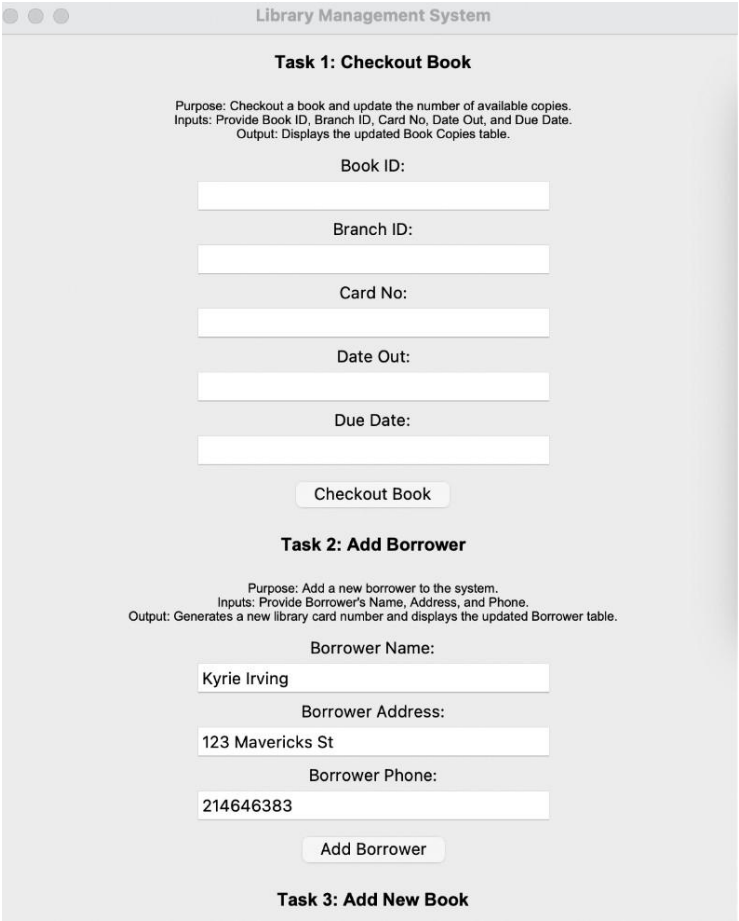
How it looks when inputting values



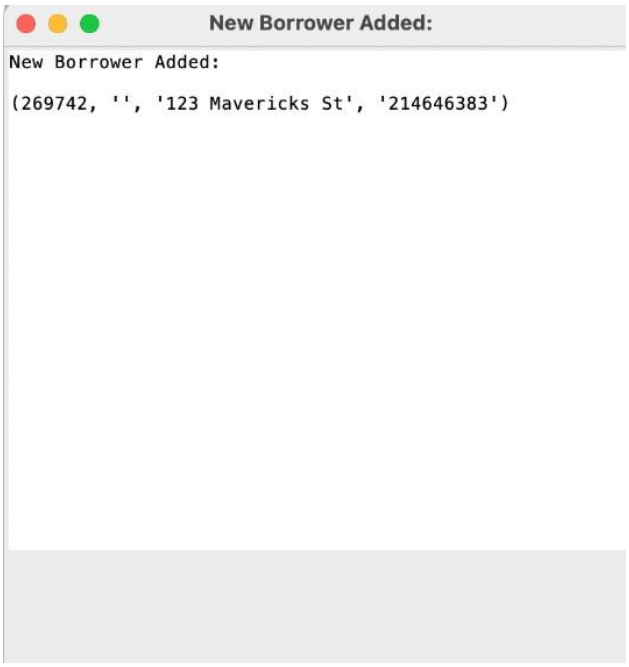
Result in GUI

(There was originally 5 copies in original database of book ID 4, after borrower with the card No 121212 checks book ID 4 out, the amount of book copies drops down to 4 - date out and due date is empty, our group just didn't insert inputs for that, regardless if we did or didn't, information for those respective inputs would show up when displaying information of the data after inputs are run)

2. Add information about a new Borrower. Do not provide the CardNo in your query. Output the card number as if you are giving a new library card. Submit your editable SQL query that your code executes.



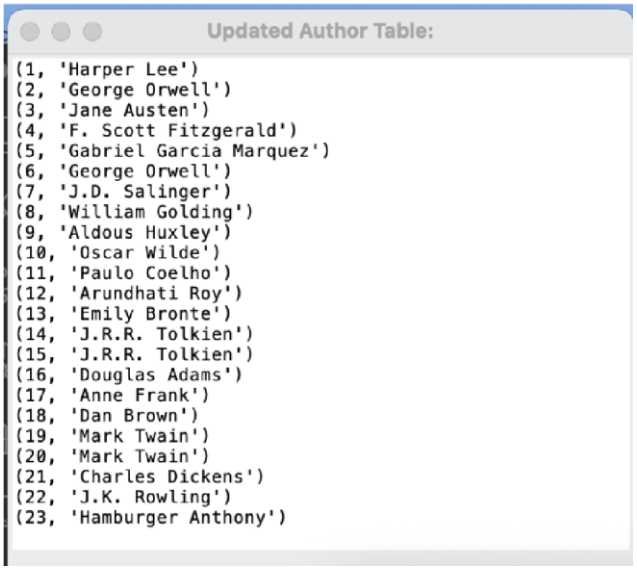
Input for new Borrower.



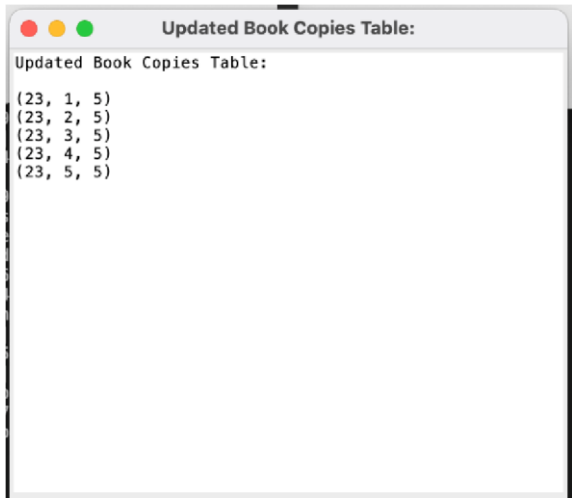
Output for Kyrie Irving being added to the borrower table along with a generated card number.

3. Add a new Book with publisher (use can use a publisher that already exists) and author information to all 5 branches with 5 copies for each branch. Submit your editable SQL query that your code executes.

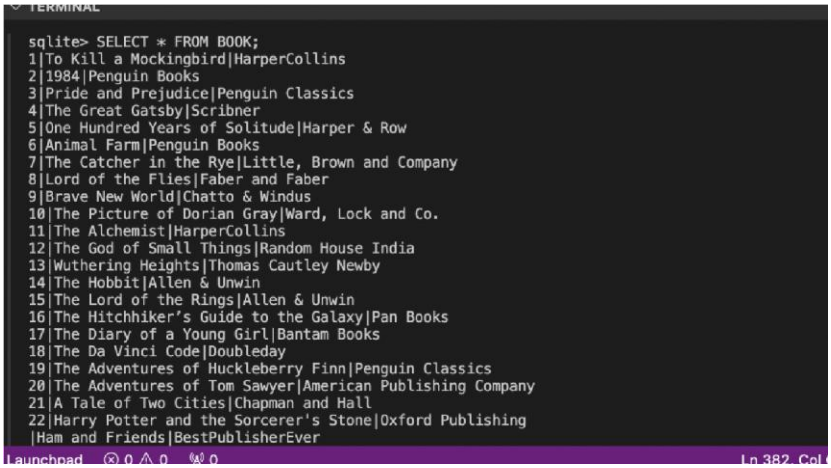
All 3 inputs entered for Step 3.



Output for Author table in GUI (shows Hamburger Anthony entered in as an author)



Output for book copies table in GUI (shows book ID 23 aka Ham and Friends have 5 copies in each in each branch (5 branches)).



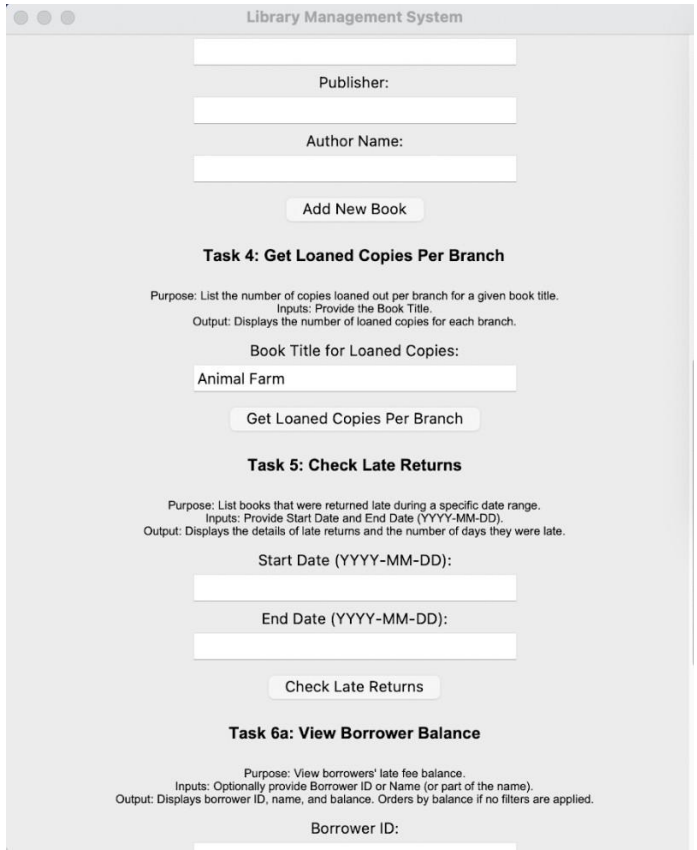
Output of book table in terminal (Ham and Friends shown as added book).

Submit your editable SQL query that your code executes.

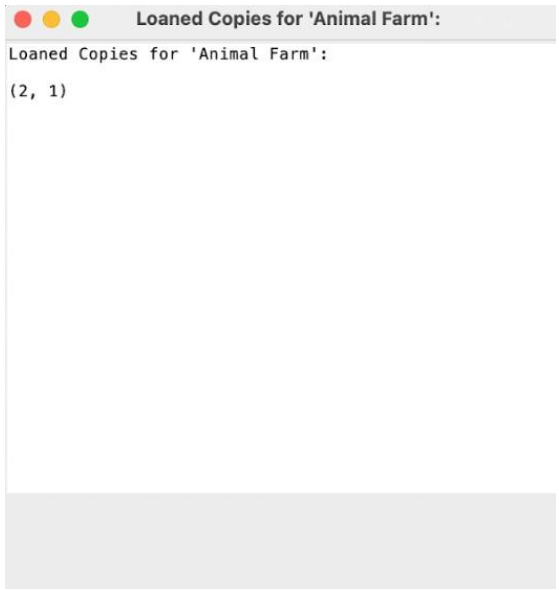
```
cursor.execute("""
        INSERT INTO Book (Title, Publisher)
        VALUES (?, ?)
        """, (book_title, publisher))
        book_id = cursor.lastrowid


        cursor.execute("""
        INSERT INTO Author (Book_ID, Author_Name)
        VALUES (?, ?)
        """, (book_id, author_name))


        for branch_id in range(1, 6):
```

```
cursor.execute("""
INSERT INTO Book_Copies (Book_ID, Branch_ID, No_Of_Copies)
VALUES (?, ?, 5)
""", (book_id, branch_id))
```

4. Given a book title list the number of copies loaned out per branch.

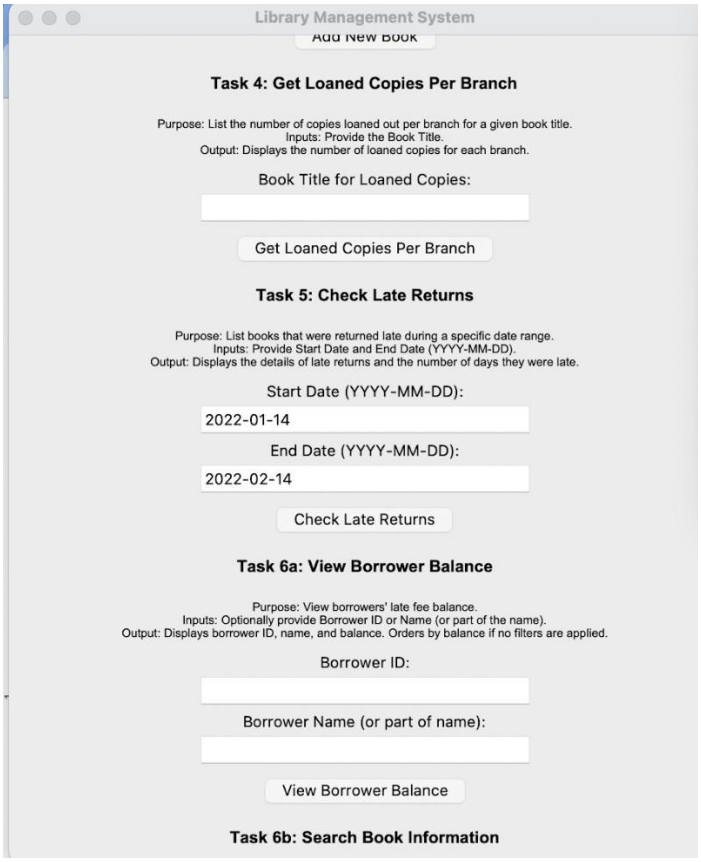


Input “ Animal Farm “ for book title entered in for step 4.



Output for Animal Farm (2 for branch 2 - West Branch and 1 copy is loaned for Animal Farm)

5. Given any due date range list the Book_Loans that were returned late and how many days they were late. Submit your editable SQL queries that your code executes.



2022-01-14 entered as an input for when a book was checked out and 2022-02-14 entered as an input of the due date of a book.

Output for all books returned past the due date (late) along with how many days they were late.

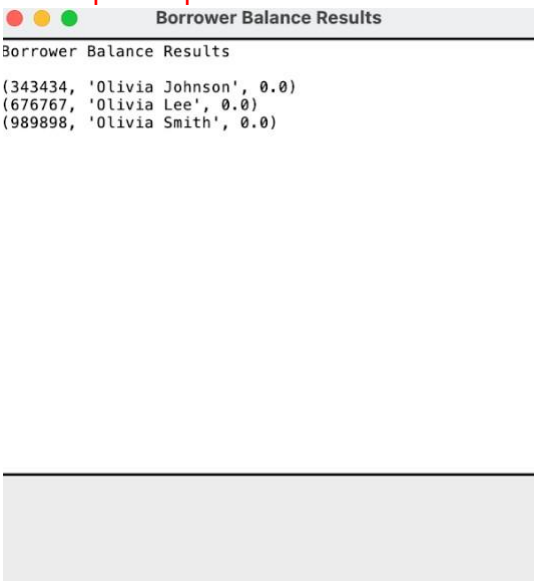
Submit your editable SQL queries that your code executes

```
cursor.execute("""
        SELECT
            bl.Book_ID,
            b.Title AS Book_Title,
            bl.Branch_ID,
            lb.Branch_Name,
            bl.Card_No,
            bl.Due_Date,
            bl.Returned_Date,
          (JULIANDAY(bl.Returned_Date) - JULIANDAY(bl.Due_Date)) AS Days_Late
        FROM
            Book_Loans bl
        JOIN
            Book b ON bl.Book_ID = b.Book_ID
        JOIN
            Library_Branch lb ON bl.Branch_ID = lb.Branch_ID
        WHERE
            bl.Returned_Date IS NOT NULL
            AND bl.Returned_Date > bl.Due_Date
            AND bl.Due_Date BETWEEN ? AND ?
        ORDER BY
            bl.Due_Date
        """, (start_date, end_date))
```

6. The fifth requirement is to return the view's results by applying the following criteria:

    a. List for every borrower the ID, name, and if there is any lateFee balance. The user has the right to search either by a borrower ID, name, part of the name, or to run the query with no filters/criteria. The amount needs to be in US dollars. For borrowers with zero (0) or NULL balance, you need to return zero dollars ($0.00). Make sure that your query returns meaningful attribute names. In the case that the user decides not to provide any filters, order the results based on the balance amount. Make sure that you return all records. Submit your editable SQL query that your code executes.

Input for part of name as " Oli "

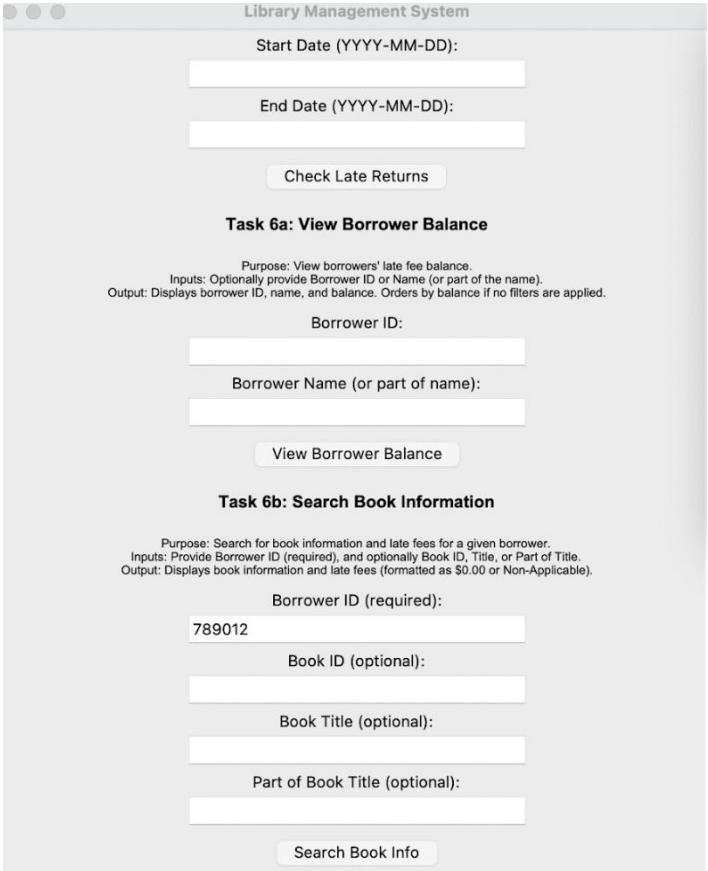


Output for all borrowers with part of name as " Oli " along with their card number along with their balance if applicable.

Submit your editable SQL query that your code executes.

```
cursor.execute("""
        SELECT
            br.Card_No AS Borrower_ID,
            br.Name AS Borrower_Name,
            IFNULL(SUM(lb.LateFee), 0.00) AS
LateFee_Balance
        FROM
            Borrower br
        LEFT JOIN
            Book_Loans bl ON br.Card_No = bl.Card_No
        LEFT JOIN
             Library_Branch lb ON bl.Branch_ID =
lb.Branch_ID
        WHERE
             (br.Card_No = ? OR ? IS NULL) AND
            (br.Name LIKE '%' || ? || '%' OR ? IS NULL)
        GROUP BY
            br.Card_No, br.Name
        ORDER BY
            LateFee_Balance DESC;
         """, (borrower_id if borrower_id else None,
               borrower_id if borrower_id else None,
               borrower_name if borrower_name else None,
                borrower_name if borrower_name else None))
```

b. List book information in the view. The user must search with borrowerID and any of the following search items: book id, books title, part of book title, or to run the query with no filters/criteria. The late fee amount needs to be in US dollars. The late fee price amount needs to have two decimals as well as the dollar '$' sign. For books that they do not have any late fee amount, you need to substitute the NULL value with a 'Non-Applicable' text. Make sure that your query returns meaningful attribute names. In the case that the user decides not to provide any filters, order the results based on the highest late fee remaining. Submit your editable SQL query that your code executes. [10 points]



Input borrower ID as 789012

Output displaying information of borrower ID 789012's book/s's information along with their borrower's name, their card number, and any late fees if applicable.

Submit your editable SQL query that your code executes.

```
query = """
       SELECT
           b.Book_ID AS Book_ID,
           b.Title AS Book_Title,
           br.Card_No AS Borrower_ID,
           br.Name AS Borrower_Name,
           CASE
               WHEN lb.LateFee IS NULL THEN 'Non-Applicable'
                ELSE '$' || printf('%.2f', lb.LateFee)
           END AS LateFee
       FROM
           Book b
       JOIN
           Book_Loans bl ON b.Book_ID = bl.Book_ID
       JOIN
           Borrower br ON bl.Card_No = br.Card_No
       LEFT JOIN
           Library_Branch lb ON bl.Branch_ID = lb.Branch_ID
       WHERE
           br.Card_No = ?
           AND (? IS NULL OR b.Book_ID = ?)
           AND (? IS NULL OR b.Title = ?)
        AND (? IS NULL OR b.Title LIKE '%' || ? || '%')
ORDER BY
    LateFee DESC;
"""
```