**Team No: Team 7**

**Class: CSE 3330**

**Semester: Fall 2023**

**Assignment: Project 2 Part 3**

**Document Date: 12/04/2023**

**Contributors:**

**Abhisekh Bajracharya**

**Chanuthi Subasinghage**

**Richard Olu Jordan**

**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.**

**TABLE OF CONTENTS**

**1. INTRODUCTION**

**2. CREATE DATABASE TABLES**

2.1 CREATE TABLE Statements

2.2 TECHNOLOGIES Used

**3. LOAD DATA INTO DATABASE TABLES**

3.1 Method to Load Data into Tables

**4. TASK 1: SQLITE TERMINAL QUERIES**

4.1 Query 1

4.2 Query 2

4.3 Query 3

**5. TASK 2: GUI INTERFACE QUERIES**

5.1 Query

5.2 Query

5.3 Query

5.4 Query

5.5 Query

5.6 Query 6a

5.7 Query 6b

**6. SQL QUERIES FOR GUI INTERFACE**

**7. TEAM CONTRIBUTIONS LIST**

**1. INTRODUCTION**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Team 7 is tasked with developing a comprehensive database for a Library Management System. This project unfolds in three distinct phases:

Phase 1 involves crafting an Entity-Relationship (ER) diagram and formulating a corresponding relational database schema.

In Phase 2, the focus shifts to creating the library management system's database in SQL. This involves executing all necessary 'CREATE TABLE' statements and integrating all constraints as specified in the relational schema.

The final stage, Phase 3, is dedicated to user interface aspects of the system. Initially, we will populate our database tables with data derived from the dataset provided in Phase 2.

This sets the stage for executing a series of SQL queries on this populated database.

The culmination of this phase is the creation of a Graphical User Interface (GUI) for the Library Management System, which will further enable us to perform additional SQL queries through this user-friendly interface.

**2.1 CREATE DATABASE TABLES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

CREATE STATEMENTS  
CREATE TABLE LIBRARY\_BRANCH (  
Branch\_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
Branch\_Name VARCHAR(25) NOT NULL,  
Address VARCHAR(50) NOT NULL  
);  
CREATE TABLE PUBLISHER(  
Name VARCHAR (50) NOT NULL,  
Phone CHAR(12) NOT NULL,  
Address VARCHAR (20) NOT NULL,  
PRIMARY KEY (Name)  
);  
CREATE TABLE BOOK(  
Book\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
Title VARCHAR(50) NOT NULL,  
Book\_Publisher VARCHAR(50) NOT NULL,  
FOREIGN KEY (Book\_Publisher) REFERENCES PUBLISHER(Name)  
);  
CREATE TABLE BORROWER (  
Card\_No INTEGER PRIMARY KEY AUTOINCREMENT CHECK(length(Card\_No) = 6),  
Name VARCHAR(25) NOT NULL,  
Address VARCHAR(25) NOT NULL,  
Phone CHAR(12) NOT NULL  
);  
CREATE TABLE BOOK\_COPIES(

Book\_id INT NOT NULL,  
Branch\_id INT NOT NULL,  
No\_of\_copies INT NOT NULL,  
PRIMARY KEY(Book\_id,Branch\_id),  
FOREIGN KEY(Book\_id) REFERENCES BOOK(Book\_id),  
FOREIGN KEY(Branch\_id) REFERENCES LIBRARY\_BRANCH(Branch\_id)  
);  
CREATE TABLE BOOK\_LOANS  
(  
Book\_id INT NOT NULL,  
Branch\_id INT NOT NULL,  
Card\_no INT NOT NULL,  
Date\_out date NOT NULL,  
Due\_date date NOT NULL,  
Returned\_date date,  
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)  
);

CREATE TABLE BOOK\_AUTHOR(  
Book\_id INTEGER PRIMARY KEY AUTOINCREMENT,  
Author\_name VARCHAR (50) NOT NULL,  
FOREIGN KEY (Book\_id) REFERENCES BOOK(Book\_id)  
);

**2.2 TECHNOLOGIES USED**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

MS WORD

SQLite Studio

Lucid Chart

DataGrip

SQLite3 database system

SQL programming language

Python 3 programming language

Linux-based operating system terminal

**3.1 LOAD DATA INTO DATABASE TABLES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

We populated the table with the use of the import statements in SQLite studio. Highlighted below are the steps:

Step 1

A screenshot of a computer

Description automatically generated

Step 2

A screenshot of a computer

Description automatically generated

Step 3

A screenshot of a computer

Description automatically generated

**4. TASK 1: SQLITE TERMINAL QUERIES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**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 have a return date later than the due date and with '0' for those were returned on time.

**Solution**

ALTER TABLE BOOK\_LOANS

ADD Late INT;

UPDATE BOOK\_LOANS

SET Late = CASE WHEN julianday(BOOK\_LOANS.Returned\_date) - julianday(BOOK\_LOANS.Date\_out) > julianday(BOOK\_LOANS.Due\_date) - julianday(BOOK\_LOANS.Date\_out) THEN 1 ELSE 0 END;

SELECT \* FROM Book\_Loans;

A screenshot of a computer

Description automatically generated

Total rows loaded: A close up of a text

Description automatically generated

**Query 2.**

Add an extra column ‘LateFee’ to the Library\_Branch table, decide late fee per day for each branch and update that column.

**Solution**

`

ALTER TABLE LIBRARY\_BRANCH

ADD LateFee DECIMAL(10,2);

UPDATE LIBRARY\_BRANCH

SET Latefee = CASE WHEN LIBRARY\_BRANCH.Branch\_Name = 'Main Branch' THEN 0.5

WHEN LIBRARY\_BRANCH.Branch\_Name = 'West Branch' THEN 2.5

WHEN LIBRARY\_BRANCH.Branch\_Name = 'East Branch' THEN 1.0

WHEN LIBRARY\_BRANCH.Branch\_Name = 'North Branch' THEN 2.0

WHEN LIBRARY\_BRANCH.Branch\_Name = 'UTA Branch' THEN 1.0

ELSE 0 END;

SELECT \* FROM LIBRARY\_BRANCH;

A screenshot of a computer

Description automatically generated

Total Rows Loaded: A black text on a white background

Description automatically generated

**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'– you 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 retuned late than fee = ‘0’

CREATE VIEW [vBookLoanInfo] AS

SELECT

BL.Book\_id,

BL.Card\_No,

BL.Date\_out,

BL.Due\_date,

BL.Returned\_date,

SUM(julianday(BL.Returned\_date) - julianday(BL.Date\_out)) AS TotalDaysLoanedOut,

B.Title,

SUM(CASE

WHEN julianday(BL.Returned\_date) > julianday(BL.Due\_date)

THEN julianday(BL.Returned\_date) - julianday(BL.Due\_date)

ELSE 0

END) AS TotalDaysReturnedLate,

BL.Branch\_id,

SUM(CASE

WHEN julianday(BL.Returned\_date) > julianday(BL.Due\_date)

THEN LB.LateFee \* (julianday(BL.Returned\_date) - julianday(BL.Due\_date))

ELSE 0

END) AS LateFeeBalance

FROM

BOOK\_LOANS BL

JOIN LIBRARY\_BRANCH LB ON LB.Branch\_ID = BL.Branch\_id

JOIN BOOK B ON B.Book\_id = BL.Book\_id

GROUP BY

BL.Book\_id, BL.Card\_No, BL.Branch\_id;

**Results From View**

A screenshot of a computer

Description automatically generated

Total Rows Loaded: 

**5. TASK 2: GUI INTERFACE QUERIES**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Query 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.

Triggers

CREATE TRIGGER update\_book\_loans

AFTER INSERT ON BOOK\_LOANS

FOR EACH ROW

BEGIN

UPDATE BOOK\_LOANS

SET Late = CASE WHEN julianday(BOOK\_LOANS.Returned\_date) > julianday(BOOK\_LOANS.Due\_date) THEN 1 ELSE 0 END

WHERE Card\_no = NEW.Card\_no;

END;

CREATE TRIGGER update\_num\_copies

AFTER INSERT ON BOOK\_LOANS

FOR EACH ROW

BEGIN

UPDATE BOOK\_COPIES

SET No\_of\_copies = No\_of\_copies - 1

WHERE BOOK\_ID = NEW.BOOK\_ID AND No\_of\_copies >= 0;

END;

Landing Page:

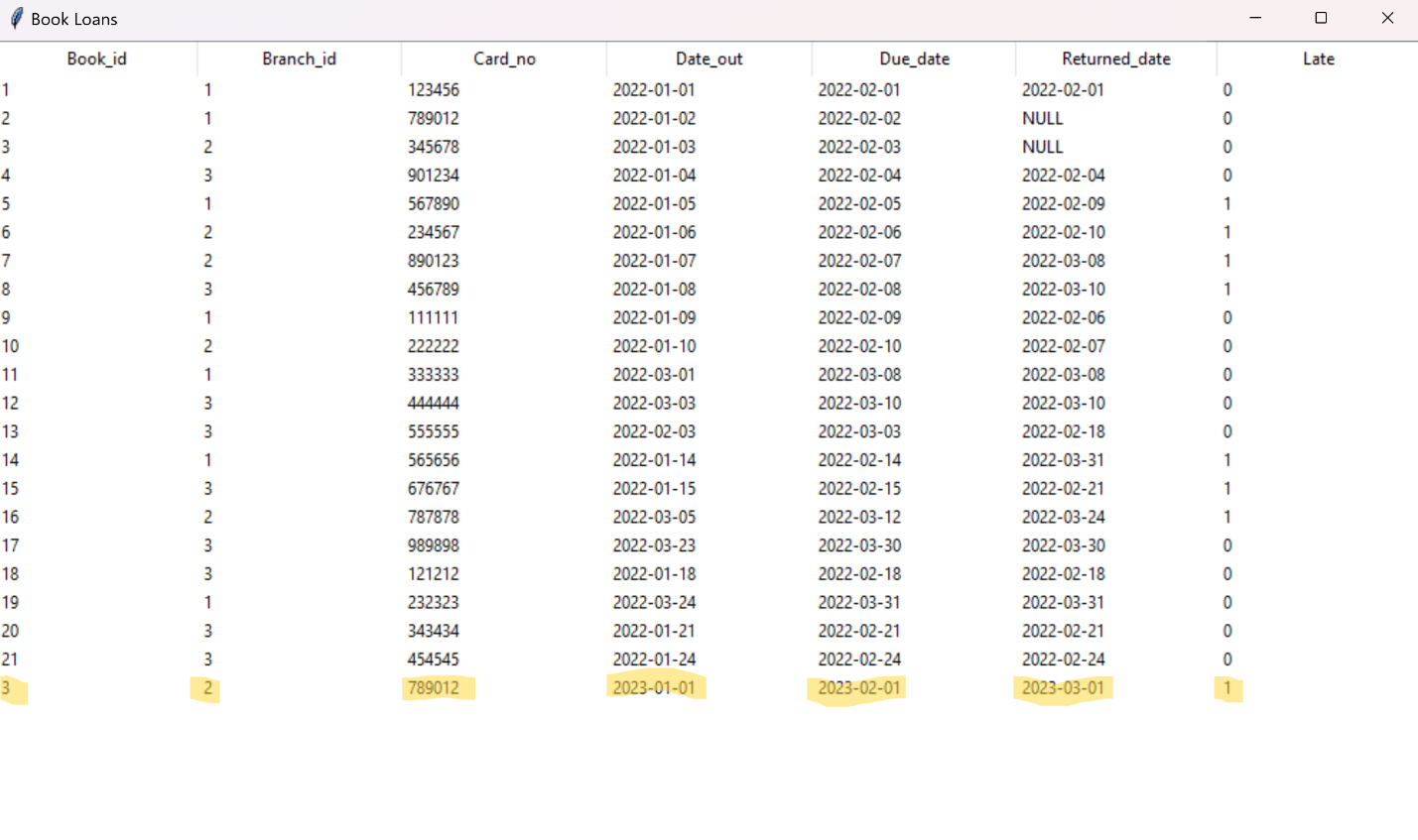


User Inputs data into the GUI:

A screenshot of a computer

Description automatically generated

User clicks on Checkout Book & Show Book Loans:



User Clicks on Show Book Copies & the value changes from 2 to 1.

A screenshot of a computer

Description automatically generated

Example 2: Different borrower checks out the same book & clicks on show book loans:

A screenshot of a computer

Description automatically generated

User clicks on Show Book Copies & the value changes from 1 to 0

A screenshot of a computer

Description automatically generated

**Query 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.

Landing Page:

A screenshot of a computer

Description automatically generated

User Enters Information:

A screenshot of a computer

Description automatically generated

User clicks on Add Borrower (borrower gets added to the system)

User clicks on Show Borrower:

A screenshot of a computer

Description automatically generated

Notes: Card\_No is autoincremented.

**Query 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.

Landing Page:

A screenshot of a computer

Description automatically generated

Notes: Book Publisher can be selected from a dropdown menu.

User Enters Information:

A screenshot of a computer

Description automatically generated

User Clicks on Show Newly Added Book:

A screenshot of a computer

Description automatically generated

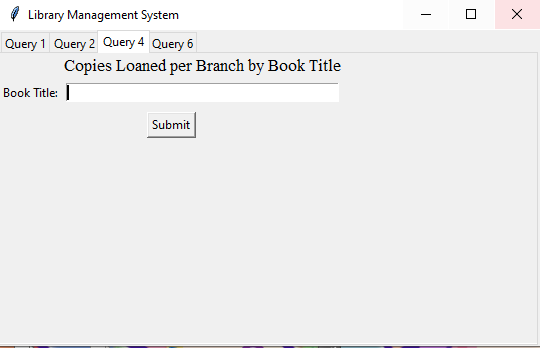
NOTES

Book\_id starts at 23 because 22 was reserved for the “Harry Potter Book” from part 2 and Book Copies was not updated at the time of database creation.

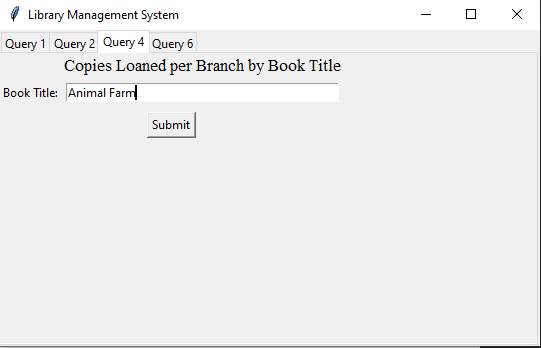
Each Branch\_ID gets the same book with 5 copies each.

Query 4:

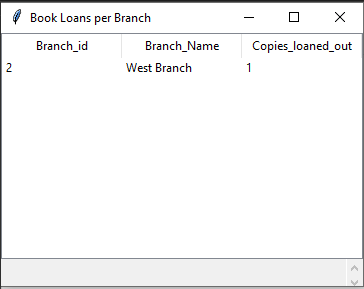
Query 4 GUI landing page



Entering Book Title:



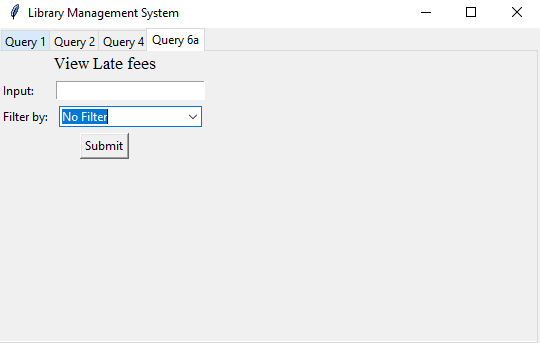
On Click Submit button: When clicked on the submit button this popup with the required information will appear



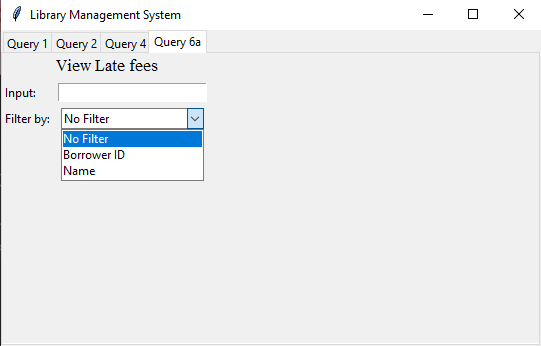
Query 6

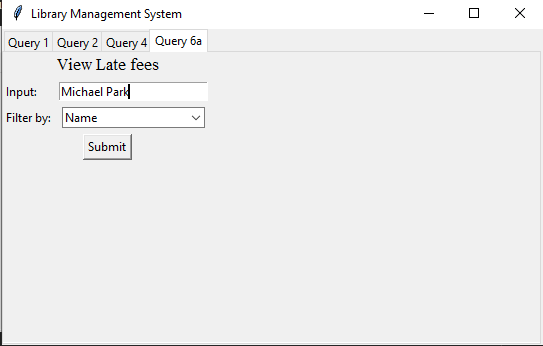
6a:

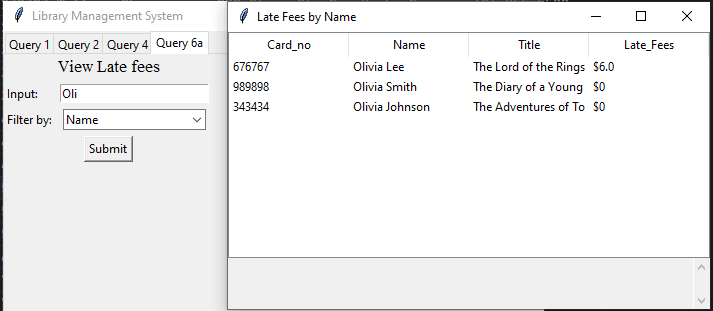
Query 6a Landing Page:



Choose any available filtering options: Type an input depending on the type of filtering if filter option is Borrower ID type Borrower ID if filter option is Name Type a Name if you don’t want any filtering click the submit button without entering any input



Example is using Name as a filtering option:   
  
Results when clicked on Submit button:  


You can also type part of the name and results will be outputted for names most like the input name:  


Query 6b:

The fifth requirement is to return the view’s results by applying the following criteria:

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.

Landing Page:

A screenshot of a computer

Description automatically generated

Example 1 User Enters Information (Borrower ID & Part of Title)

A screenshot of a computer

Description automatically generated

NOTES

User can select Borrower ID from drop down menu.

User can enter either the full Book Title, Part of Title or the book\_id and get results.

User Clicks Submit:

A screenshot of a computer

Description automatically generated

Example 2 User Enter Information (Borrower ID & Full Title)

A screenshot of a computer

Description automatically generated

User Clicks Submit:

A screenshot of a computer

Description automatically generated

Example 3 User Enter Information (Borrower ID & BookID)

A screenshot of a computer

Description automatically generated

User Clicks Submit:

A screenshot of a computer

Description automatically generated

User Can Choose to return results without filters ordered by highest late fee remaining:

Upon clicking View Book Info.

A screenshot of a computer

Description automatically generated

**6. SQL QUERIES FOR GUI INTERFACE**

**Please Refer to Code Folder.**

**In the doc entitled “project3.py”**

**ALL SQL QUERIES THAT CODE EXECUTES WILL BE LOCATED THERE.**

**7. TEAM CONTRIBUTION LIST**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Abhisekh Bajracharya

1. CREATE STATEMENTS

* LIBRARY\_BRANCH
* PUBLISHER

Chanuthi Subasinghage

1. CREATE STATEMENTS

* BOOK
* BOOK\_AUTHORS
* BOOK\_LOANS

Richard Olu Jordan

1. CREATE STATEMENTS

* BORROWER
* BOOK\_COPIES

1. TASK 1

* QUERY 3

1. TASK 2

* QUERY 1
* QUERY 2
* QUERY 6b