Library Project Management Database

TEAM 1

Hector Trevino

Kosuke Satake

Muna Bhattarai

**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.

**Task 1**

**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 that were returned on time:**

ALTER TABLE BOOK\_LOANS

ADD COLUMN Late INT;

UPDATE BOOK\_LOANS

SET Late = CASE

WHEN Returned\_date > Due\_Date THEN 1

ELSE 0

END;

A screenshot of a computer

Description automatically generated

(21 records returned, 21 records affected)

**Query 2. Add an extra column ‘LateFee’ to the Library\_Branch table, decide the late fee per day for each branch and update that column:**

ALTER TABLE LIBRARY\_BRANCH

ADD COLUMN LateFee DECIMAL(8, 2);

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 0.60

ELSE 0.00

END;

A computer screen shot of a black screen

Description automatically generated

(3 records returned, 3 records affected)

**Query 3. Create a view vBookLoanInfo that retrieves all information per book loan:**

CREATE VIEW vBookLoanInfo AS

SELECT

BL.Card\_No,

BORROWER.Name AS Borrower\_Name,

BL.Date\_Out,

BL.Due\_Date,

BL.Returned\_date,

CAST((julianday(BL.Returned\_date) - julianday(BL.Date\_Out)) AS INTEGER) AS TotalDays,

BOOK.Title AS Book\_Title,

CASE WHEN BL.Returned\_date > BL.Due\_Date THEN CAST((julianday(BL.Returned\_date) - julianday(BL.Due\_Date)) AS INTEGER) ELSE 0 END AS DaysReturnedLate,

BL.Branch\_Id,

CASE WHEN BL.Returned\_date > BL.Due\_Date THEN LIBRARY\_BRANCH.LateFee \* CAST((julianday(BL.Returned\_date) - julianday(BL.Due\_Date)) AS INTEGER) ELSE 0 END AS LateFeeBalance

FROM

BOOK\_LOANS BL

JOIN

BORROWER ON BL.Card\_No = BORROWER.Card\_No

JOIN

BOOK ON BL.Book\_Id = BOOK.Book\_Id

JOIN

LIBRARY\_BRANCH ON BL.Branch\_Id = LIBRARY\_BRANCH.Branch\_Id;

SELECT VIEWA black screen with many small colored text

Description automatically generated with medium confidence

(21 records returned, 21 records affected)

**Task 2**

**Requirements:**

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:**

Trigger.sql:

CREATE TRIGGER IF NOT EXISTS update\_book\_copies\_after\_loan

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 Branch\_Id = NEW.Branch\_Id;

END;

GUI\_Check\_Out.py:

import sqlite3

import tkinter as tk

from datetime import datetime, timedelta

from dateutil.relativedelta import relativedelta

def execute\_sql\_file(file\_path, db\_path):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

with open(file\_path, 'r') as sql\_file:

sql\_script = sql\_file.read()

cursor.executescript(sql\_script)

conn.commit()

conn.close()

def checkout\_book(db\_path, book\_id, branch\_id, card\_no, date\_out, due\_date):

conn = sqlite3.connect(db\_path)

cursor = conn.cursor()

cursor.execute("""

INSERT INTO BOOK\_LOANS (Book\_Id, Branch\_Id, Card\_No, Date\_Out, Due\_Date)

VALUES (?, ?, ?, ?, ?)

""", (book\_id, branch\_id, card\_no, date\_out, due\_date))

conn.commit()

conn.close()

def on\_checkout():

# Calculate due date as one month from date\_out

date\_out = datetime.strptime(date\_out\_entry.get(), "%Y-%m-%d")

due\_date = date\_out + relativedelta(months=1)

# Perform checkout

checkout\_book(

db\_path=db\_path,

book\_id=int(book\_id\_entry.get()),

branch\_id=int(branch\_id\_entry.get()),

card\_no=int(card\_no\_entry.get()),

date\_out=date\_out.strftime("%Y-%m-%d"),

due\_date=due\_date.strftime("%Y-%m-%d")

)

# Confirmation message

result\_label.config(text="Book checked out successfully.")

# Path to your SQLite database

db\_path = 'CSE3330.db'

# Execute the trigger creation script

execute\_sql\_file('1.Trigger.sql', db\_path)

# Tkinter GUI setup

root = tk.Tk()

root.title("Book Checkout")

tk.Label(root, text="Book ID:").pack()

book\_id\_entry = tk.Entry(root)

book\_id\_entry.pack()

tk.Label(root, text="Branch ID:").pack()

branch\_id\_entry = tk.Entry(root)

branch\_id\_entry.pack()

tk.Label(root, text="Card No:").pack()

card\_no\_entry = tk.Entry(root)

card\_no\_entry.pack()

tk.Label(root, text="Date Out (YYYY-MM-DD):").pack()

date\_out\_entry = tk.Entry(root)

date\_out\_entry.pack()

checkout\_button = tk.Button(root, text="Checkout", command=on\_checkout)

checkout\_button.pack()

result\_label = tk.Label(root, text="")

result\_label.pack()

root.mainloop()

GUI\_BC.py:

import sqlite3

import tkinter as tk

from tkinter import ttk

# Connect to your database

connection = sqlite3.connect('CSE3330.db')

# Query the BOOK\_COPIES table

query = "SELECT \* FROM BOOK\_COPIES"

cursor = connection.cursor()

cursor.execute(query)

rows = cursor.fetchall()

# Set up the GUI

root = tk.Tk()

root.title("Book Copies")

# Create a treeview to display the data

tree = ttk.Treeview(root, columns=('Book\_Id', 'Branch\_Id', 'No\_Of\_Copies'), show='headings')

tree.heading('Book\_Id', text='Book ID')

tree.heading('Branch\_Id', text='Branch ID')

tree.heading('No\_Of\_Copies', text='Number of Copies')

# Insert the data into the treeview

for row in rows:

tree.insert('', tk.END, values=row)

tree.pack(expand=True, fill='both')

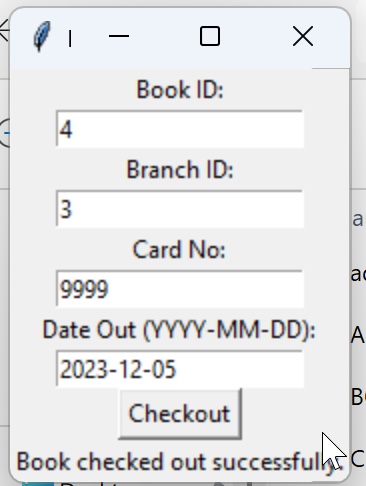
# Start the GUI loop

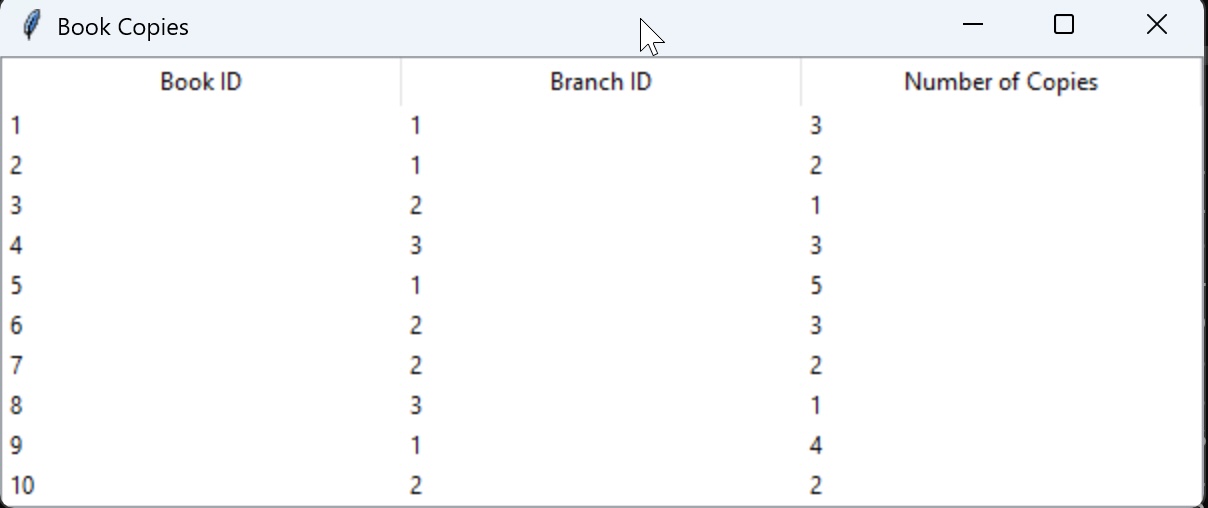
root.mainloop()

# Close the database connection

connection.close()

Screenshot:





1. **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:**

2.Add\_BORROWER.sql:

INSERT INTO BORROWER (Name, Address, Phone)

VALUES (?, ?, ?);

GUI\_Borrower.py:

import sqlite3

import tkinter as tk

from tkinter import messagebox

def insert\_borrower(name, address, phone, file\_path='2.Add\_BORROWER.sql'):

try:

# Read SQL command from the file

with open(file\_path, 'r') as file:

sql\_command = file.read()

# Connect to the SQLite database

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Execute the SQL command with the provided parameters

cursor.execute(sql\_command, (name, address, phone))

conn.commit()

# Retrieve the last inserted row ID (card number)

card\_number = cursor.lastrowid

messagebox.showinfo("Success", f"Borrower added successfully!\nCard Number: {card\_number}")

except sqlite3.Error as error:

messagebox.showerror("Error", f"Error occurred: {error}")

finally:

conn.close()

def on\_submit():

name = name\_entry.get()

address = address\_entry.get()

phone = phone\_entry.get()

insert\_borrower(name, address, phone)

# GUI Setup

root = tk.Tk()

root.title("Add Borrower")

# Labels and Entry Widgets

tk.Label(root, text="Name").grid(row=0, column=0)

name\_entry = tk.Entry(root)

name\_entry.grid(row=0, column=1)

tk.Label(root, text="Address").grid(row=1, column=0)

address\_entry = tk.Entry(root)

address\_entry.grid(row=1, column=1)

tk.Label(root, text="Phone").grid(row=2, column=0)

phone\_entry = tk.Entry(root)

phone\_entry.grid(row=2, column=1)

# Submit Button

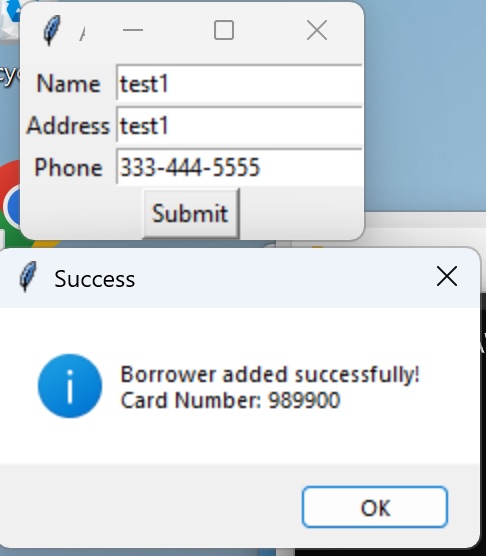
submit\_button = tk.Button(root, text="Submit", command=on\_submit)

submit\_button.grid(row=3, column=0, columnspan=2)

# Run the application

root.mainloop()

Screenshot:



1. **Add a new Book with publisher (can use a publisher that already exists) and author information to all 5 branches with 5 copies for each branch:**

3.Add\_BOOK.sql:

-- Insert a new book

INSERT INTO BOOK (Title, Publisher\_name) VALUES (?, ?);

-- Insert author information

INSERT INTO BOOK\_AUTHORS (Book\_Id, Author\_Name) VALUES (?, ?);

-- Insert book copies into each branch

INSERT INTO BOOK\_COPIES (Book\_Id, Branch\_Id, No\_Of\_Copies) VALUES (?, ?, ?);

GUI\_ADD\_BOOK.py:

import sqlite3

import tkinter as tk

from tkinter import messagebox

def add\_book(title, publisher\_name, author\_name, file\_path='3.Add\_BOOK.sql'):

try:

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Read SQL commands from the file

with open(file\_path, 'r') as file:

# Assuming the file has the queries in the correct order and format

insert\_book\_query, insert\_author\_query, insert\_copies\_query = file.read().split(';')[:-1]

# Insert the book and get the book\_id

cursor.execute(insert\_book\_query, (title, publisher\_name))

book\_id = cursor.lastrowid

# Insert author information

cursor.execute(insert\_author\_query, (book\_id, author\_name))

# Insert book copies into each branch

for branch\_id in range(1, 6):

cursor.execute(insert\_copies\_query, (book\_id, branch\_id, 5))

conn.commit()

messagebox.showinfo("Success", "Book added successfully to all branches!")

except sqlite3.Error as error:

conn.rollback()

messagebox.showerror("Error", f"Error occurred: {error}")

finally:

conn.close()

def on\_submit():

title = title\_entry.get()

publisher\_name = publisher\_entry.get()

author\_name = author\_entry.get()

add\_book(title, publisher\_name, author\_name)

# GUI Setup

root = tk.Tk()

root.title("Add Book to Branches")

# Labels and Entry Widgets

tk.Label(root, text="Title").grid(row=0, column=0)

title\_entry = tk.Entry(root)

title\_entry.grid(row=0, column=1)

tk.Label(root, text="Publisher Name").grid(row=1, column=0)

publisher\_entry = tk.Entry(root)

publisher\_entry.grid(row=1, column=1)

tk.Label(root, text="Author Name").grid(row=2, column=0)

author\_entry = tk.Entry(root)

author\_entry.grid(row=2, column=1)

# Submit Button

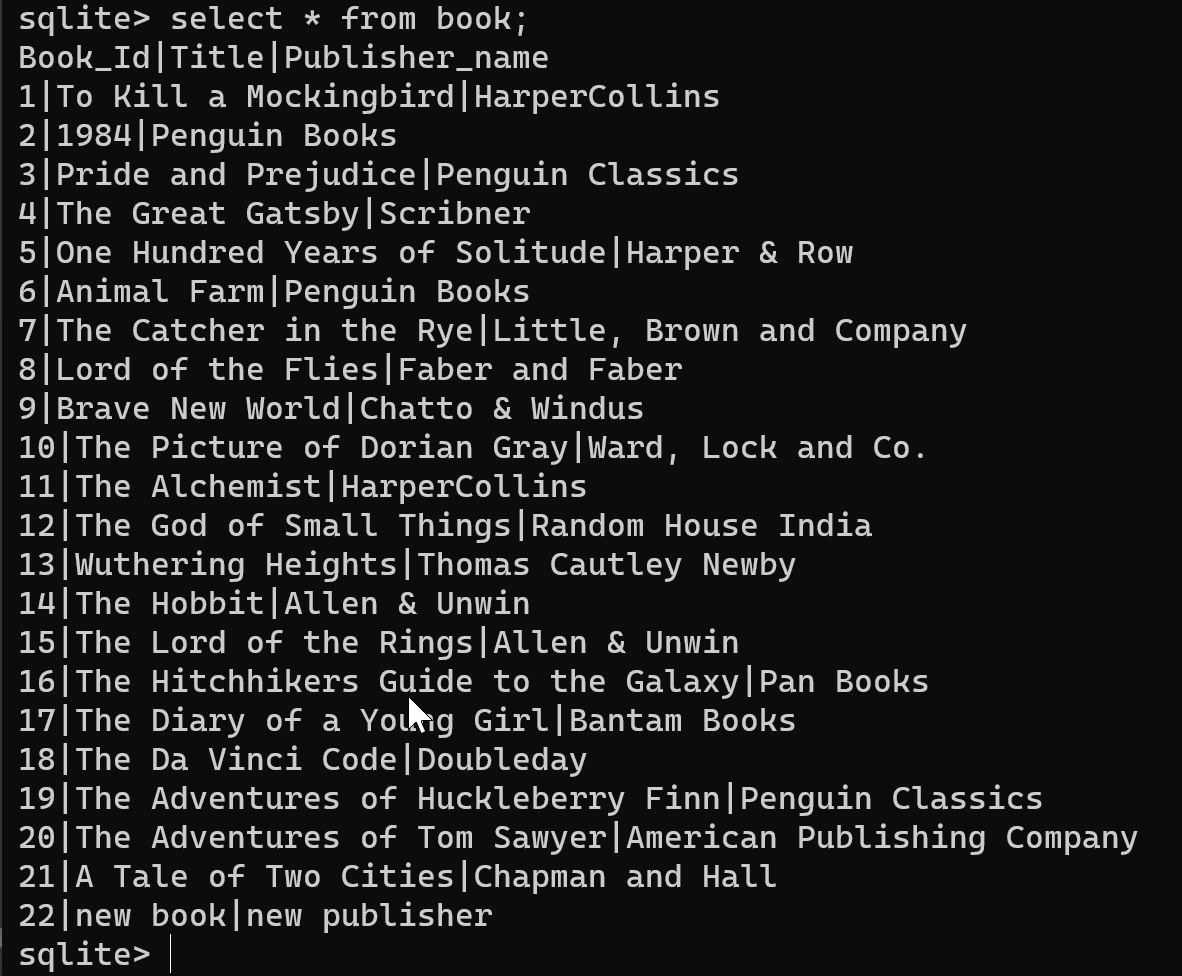
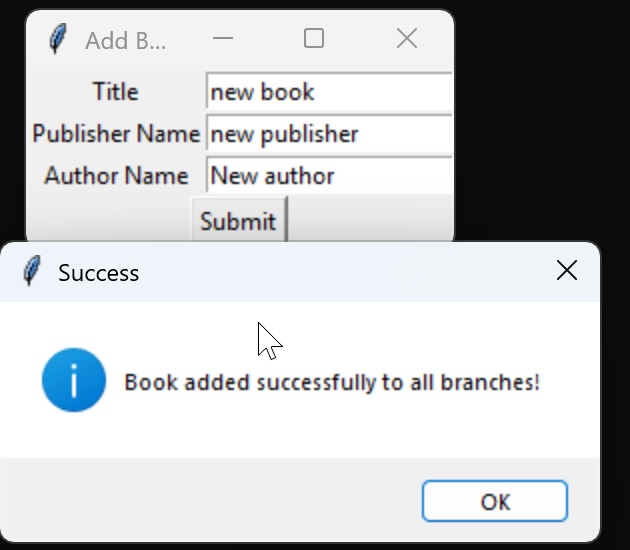
submit\_button = tk.Button(root, text="Submit", command=on\_submit)

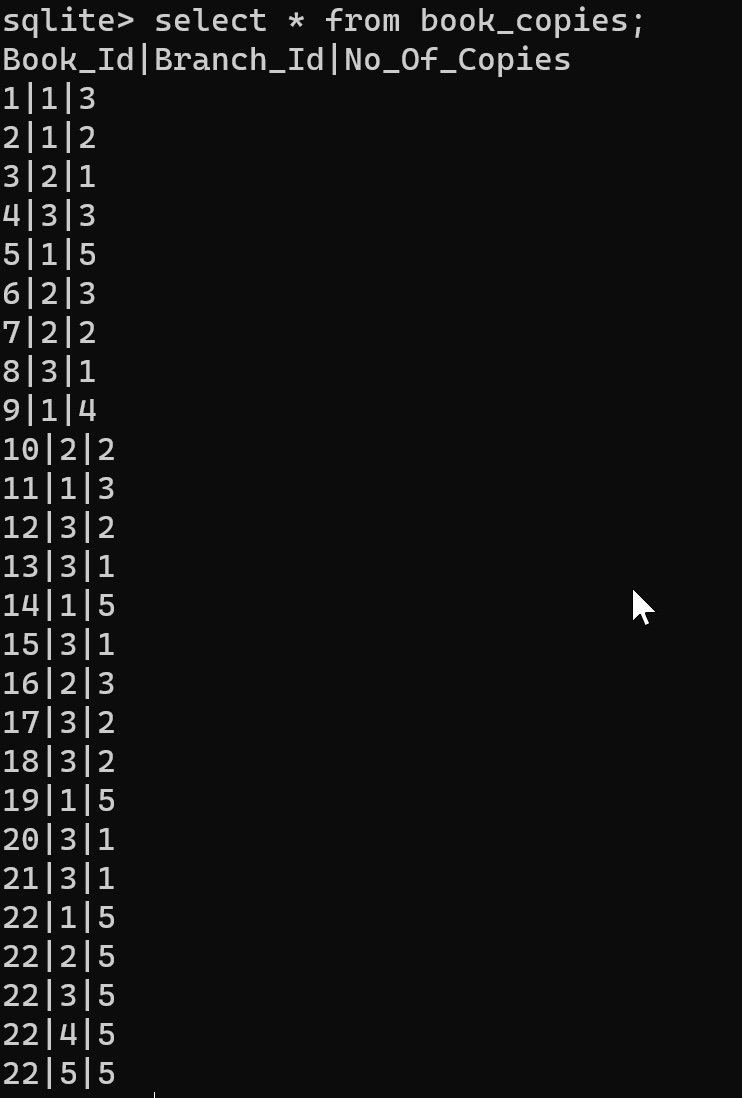
submit\_button.grid(row=3, column=0, columnspan=2)

# Run the application

root.mainloop()

Screenshot:





1. **Given a book title list the number of copies loaned out per branch:**

GUI\_No\_of\_Copies.py:

import tkinter as tk

import sqlite3

def find\_loaned\_out\_copies(book\_title):

# Connect to your SQLite database

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Updated SQL query to find loaned out copies per branch

query = """

SELECT LB.Branch\_Name, COUNT(\*) as Loaned\_Out\_Copies

FROM BOOK B

INNER JOIN BOOK\_LOANS BL ON B.Book\_Id = BL.Book\_Id

INNER JOIN LIBRARY\_BRANCH LB ON BL.Branch\_Id = LB.Branch\_Id

LEFT JOIN BOOK\_COPIES BC ON B.Book\_Id = BC.Book\_Id AND LB.Branch\_Id = BC.Branch\_Id

WHERE B.Title = ? AND BL.Returned\_date IS NULL

GROUP BY LB.Branch\_Name;

"""

# Execute the query

cursor.execute(query, (book\_title,))

results = cursor.fetchall()

# Close the database connection

conn.close()

return results

def on\_search():

book\_title = title\_entry.get()

results = find\_loaned\_out\_copies(book\_title)

result\_text = '\n'.join([f"{branch}: {count} copies" for branch, count in results])

result\_label.config(text=result\_text)

# Set up the GUI

root = tk.Tk()

root.title("Book Loan Search")

tk.Label(root, text="Enter Book Title:").pack()

title\_entry = tk.Entry(root)

title\_entry.pack()

search\_button = tk.Button(root, text="Search", command=on\_search)

search\_button.pack()

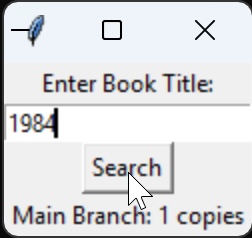
result\_label = tk.Label(root, text="", justify=tk.LEFT)

result\_label.pack()

# Start the GUI loop

root.mainloop()

Screenshot:



1. **Given any due date range list the Book\_Loans that were returned late and how many days they were late:**

5.late\_returns.sql:

SELECT BL.Book\_Id, BL.Branch\_Id, BL.Card\_No, BL.Due\_Date, BL.Returned\_date,

JULIANDAY(BL.Returned\_date) - JULIANDAY(BL.Due\_Date) AS Days\_Late

FROM BOOK\_LOANS BL

WHERE BL.Returned\_date > BL.Due\_Date

AND BL.Due\_Date BETWEEN ? AND ?

GUI\_Due\_Date\_Range.py:

import tkinter as tk

import sqlite3

from datetime import datetime

def find\_late\_returns(start\_date, end\_date):

# Connect to your SQLite database

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Read the SQL query from the file

with open('5.late\_returns.sql', 'r') as file:

query = file.read()

# Execute the query

cursor.execute(query, (start\_date, end\_date))

results = cursor.fetchall()

# Close the database connection

conn.close()

return results

# Execute the query

cursor.execute(query, (start\_date, end\_date))

results = cursor.fetchall()

# Close the database connection

conn.close()

return results

def on\_search():

start\_date = start\_date\_entry.get()

end\_date = end\_date\_entry.get()

results = find\_late\_returns(start\_date, end\_date)

result\_text = '\n'.join([f"Book ID: {book\_id}, Branch ID: {branch\_id}, Card No: {card\_no}, Due: {due}, Returned: {returned}, Days Late: {days\_late}"

for book\_id, branch\_id, card\_no, due, returned, days\_late in results])

result\_label.config(text=result\_text)

# Set up the GUI

root = tk.Tk()

root.title("Late Book Returns Search")

tk.Label(root, text="Enter Start Date (YYYY-MM-DD):").pack()

start\_date\_entry = tk.Entry(root)

start\_date\_entry.pack()

tk.Label(root, text="Enter End Date (YYYY-MM-DD):").pack()

end\_date\_entry = tk.Entry(root)

end\_date\_entry.pack()

search\_button = tk.Button(root, text="Search", command=on\_search)

search\_button.pack()

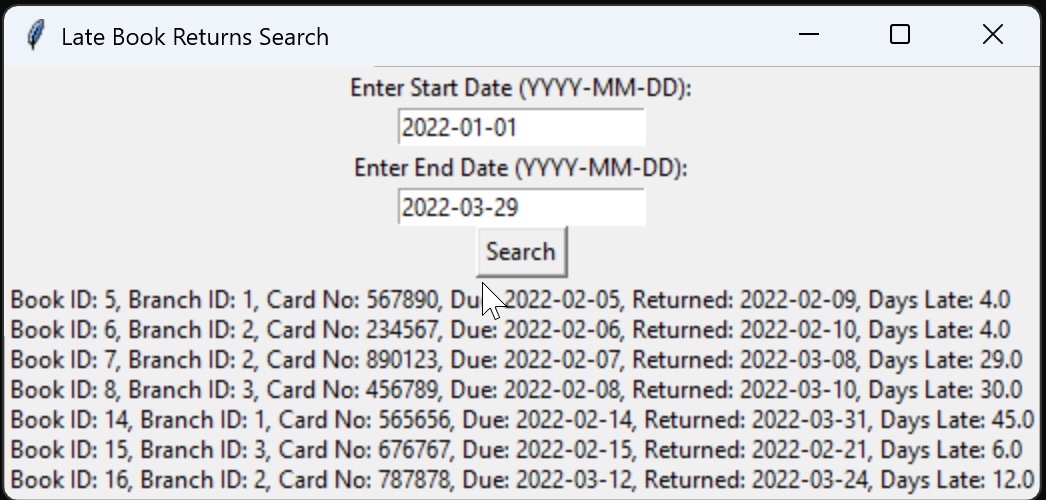
result\_label = tk.Label(root, text="", justify=tk.LEFT)

result\_label.pack()

# Start the GUI loop

root.mainloop()

Screenshot:



**6a. 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):**

import tkinter as tk

from tkinter import ttk

import sqlite3

def execute\_query(user\_id=None, user\_name=None):

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Read the SQL query from the file

with open('6.List\_for\_Every\_Borrower.sql', 'r') as file:

query = file.read()

cursor.execute(query, (user\_id, user\_name, user\_id, user\_name))

results = cursor.fetchall()

conn.close()

return results

def run\_query():

user\_id\_input = entry\_id.get()

user\_name\_input = entry\_name.get()

results = execute\_query(user\_id\_input or None, user\_name\_input or None)

# Clear the existing treeview content

for row in tree.get\_children():

tree.delete(row)

# Populate the treeview with new results

for result in results:

tree.insert('', 'end', values=result)

# GUI setup

root = tk.Tk()

root.title("Borrower Query GUI")

# Entry fields for user input

label\_id = tk.Label(root, text="Borrower ID:")

label\_name = tk.Label(root, text="Borrower Name:")

entry\_id = tk.Entry(root)

entry\_name = tk.Entry(root)

# Button to run the query

query\_button = tk.Button(root, text="Run Query", command=run\_query)

# Treeview to display results

tree\_columns = ("Borrower ID", "Borrower Name", "Late Fee Balance")

tree = ttk.Treeview(root, columns=tree\_columns, show="headings")

for col in tree\_columns:

tree.heading(col, text=col)

tree.column(col, anchor="center", width=100)

# Pack widgets

label\_id.grid(row=0, column=0, padx=10, pady=5, sticky="e")

entry\_id.grid(row=0, column=1, padx=10, pady=5)

label\_name.grid(row=1, column=0, padx=10, pady=5, sticky="e")

entry\_name.grid(row=1, column=1, padx=10, pady=5)

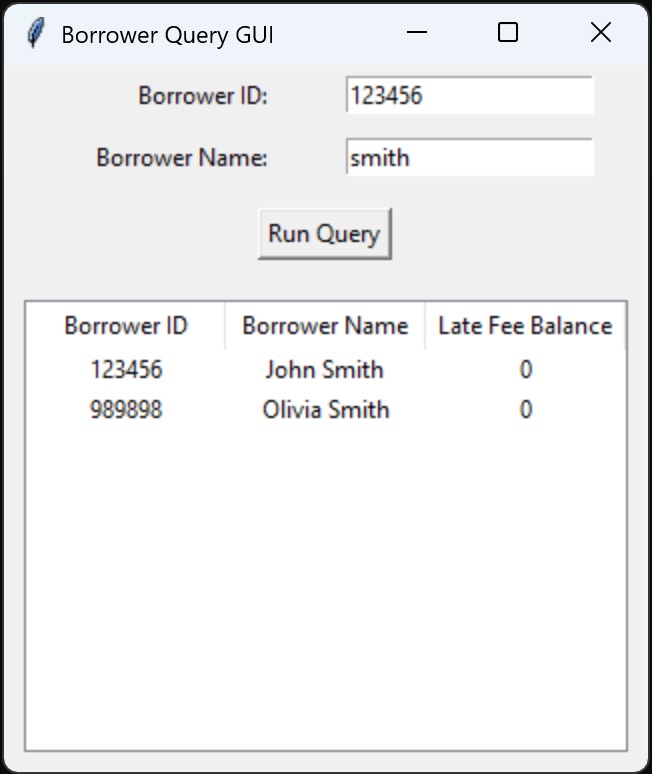
query\_button.grid(row=2, column=0, columnspan=2, pady=10)

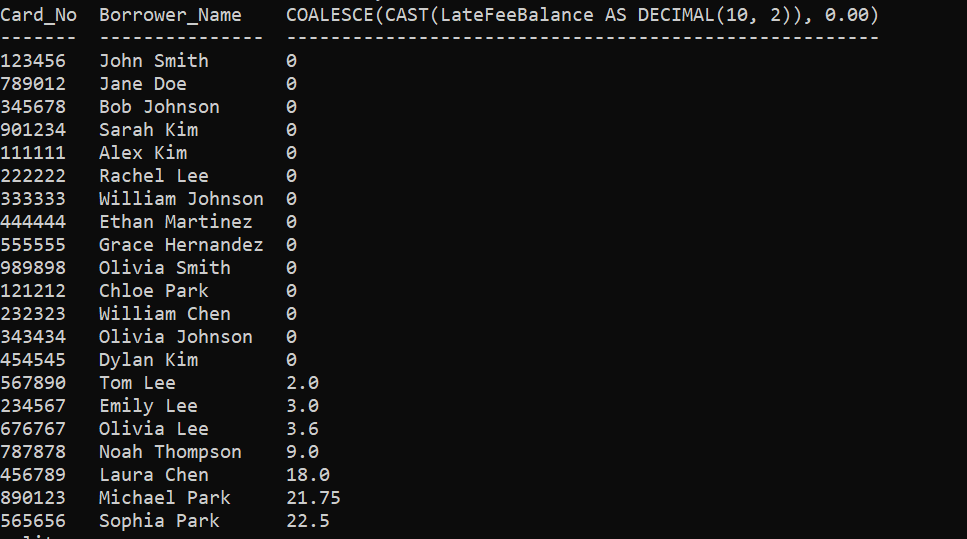
tree.grid(row=3, column=0, columnspan=2, padx=10, pady=10)

# Run the GUI

root.mainloop()

Screenshot:





**6b. 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:**

from tkinter import \*

import sqlite3

from tkinter import ttk

def create\_view():

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Read SQL query from an external file

with open('6ba.sql', 'r') as file:

sql\_query = file.read()

# Create the view

cursor.execute(sql\_query)

# Commit the changes and close the connection

conn.commit()

conn.close()

def execute\_query(Card\_No=None, Book\_Id=None, Title=None):

conn = sqlite3.connect('CSE3330.db')

cursor = conn.cursor()

# Read SQL query from an external file

with open('6bb.sql', 'r') as file:

sql\_query = file.read()

# Add the WHERE clause dynamically to the SQL query

sql\_query += f"""

WHERE

(

Card\_No = ? OR

Book\_Id = ? OR

Title LIKE '%' || ? || '%' OR

? IS NULL AND ? IS NULL AND ? IS NULL

)

ORDER BY

LateFee DESC NULLS LAST;

"""

cursor.execute(sql\_query, (Card\_No, Book\_Id, Title, Card\_No, Book\_Id, Title))

results = cursor.fetchall()

conn.close()

return results

def run\_query():

Card\_No\_input = entry\_Card\_No.get()

Book\_Id\_input = entry\_Book\_Id.get()

Title\_input = entry\_Title.get()

results = execute\_query(Card\_No\_input or None, Book\_Id\_input or None, Title\_input or None)

# Clear the existing treeview content

for row in tree.get\_children():

tree.delete(row)

# Populate the treeview with new results

for result in results:

tree.insert('', 'end', values=result)

# GUI setup

root = Tk()

root.title("Book Query GUI")

# Entry fields for user input

label\_Card\_No = Label(root, text="Borrower ID:")

label\_Book\_Id = Label(root, text="Book ID:")

label\_Title = Label(root, text="Book Title:")

entry\_Card\_No = Entry(root)

entry\_Book\_Id = Entry(root)

entry\_Title = Entry(root)

# Button to create view and run the query

create\_view\_button = Button(root, text="Create View", command=create\_view)

query\_button = Button(root, text="Run Query", command=run\_query)

# Treeview to display results

tree\_columns = ("Borrower ID", "Book ID", "Book Title", "Late Fee Amount")

tree = ttk.Treeview(root, columns=tree\_columns, show="headings")

for col in tree\_columns:

tree.heading(col, text=col)

tree.column(col, anchor="center", width=100)

# Pack widgets

label\_Card\_No.grid(row=0, column=0, padx=10, pady=5, sticky="e")

entry\_Card\_No.grid(row=0, column=1, padx=10, pady=5)

label\_Book\_Id.grid(row=1, column=0, padx=10, pady=5, sticky="e")

entry\_Book\_Id.grid(row=1, column=1, padx=10, pady=5)

label\_Title.grid(row=2, column=0, padx=10, pady=5, sticky="e")

entry\_Title.grid(row=2, column=1, padx=10, pady=5)

create\_view\_button.grid(row=3, column=0, pady=5)

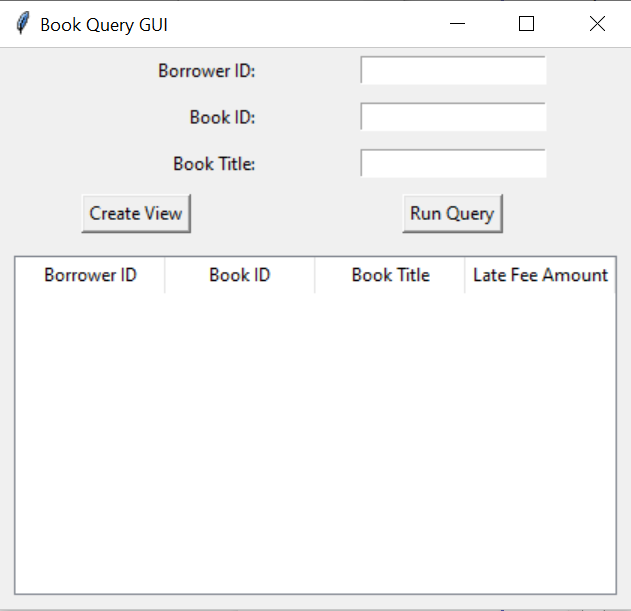
query\_button.grid(row=3, column=1, pady=5)

tree.grid(row=4, column=0, columnspan=2, padx=10, pady=10)

# Run the GUI

root.mainloop()

Screenshot:





Contribution:

Trevino Hector: Task 1, Documentation

Kosuke Satake: Task 2, Documentation

Muna Bhattarai: Task 2, Documentation