Library System

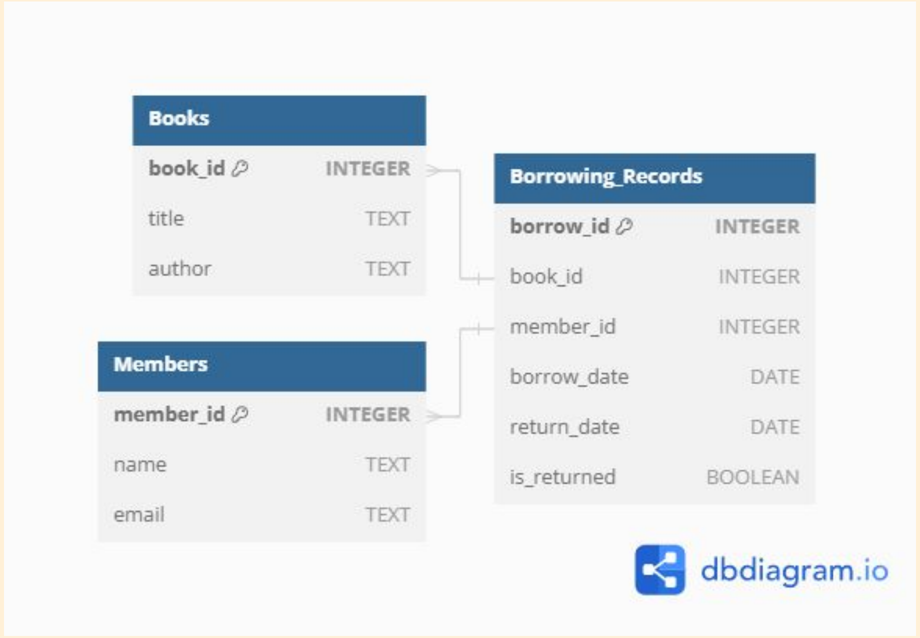
Donovan, Joyce

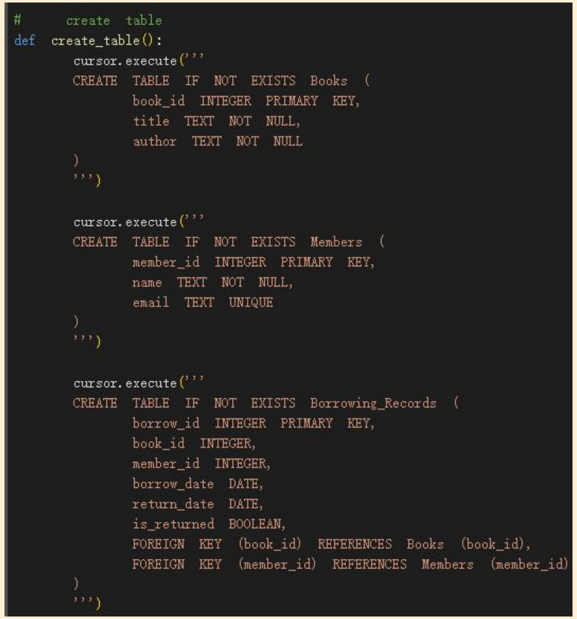
Table of contents

01

Schema

First bookstores





```
#     create  table
def  create_table():
        cursor.execute('''
        CREATE  TABLE  IF  NOT  EXISTS  Books  (
                book_id  INTEGER  PRIMARY  KEY,
                title  TEXT  NOT  NULL,
                author  TEXT  NOT  NULL
        )
        ''')

        cursor.execute('''
        CREATE  TABLE  IF  NOT  EXISTS  Members  (
                member_id  INTEGER  PRIMARY  KEY,
                name  TEXT  NOT  NULL,
                email  TEXT  UNIQUE
        )
        ''')

        cursor.execute('''
        CREATE  TABLE  IF  NOT  EXISTS  Borrowing_Records  (
                borrow_id  INTEGER  PRIMARY  KEY,
                book_id  INTEGER,
                member_id  INTEGER,
                borrow_date  DATE,
                return_date  DATE,
                is_returned  BOOLEAN,
                FOREIGN  KEY  (book_id)  REFERENCES  Books  (book_id),
                FOREIGN  KEY  (member_id)  REFERENCES  Members  (member_id)
        )
        ''')
```

02

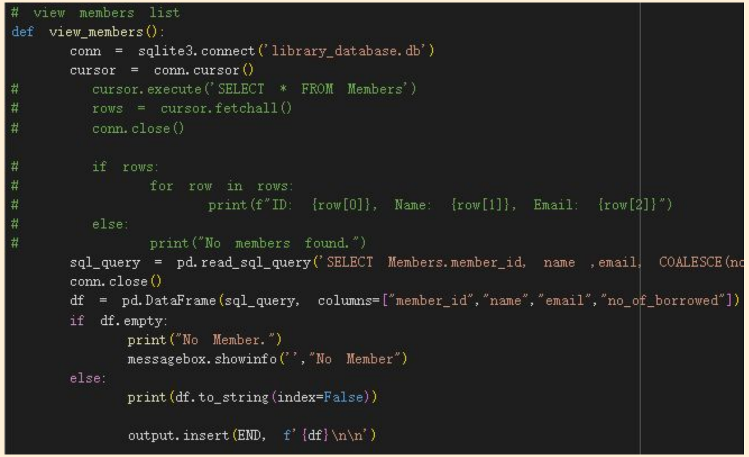
Basic Components

Back-END

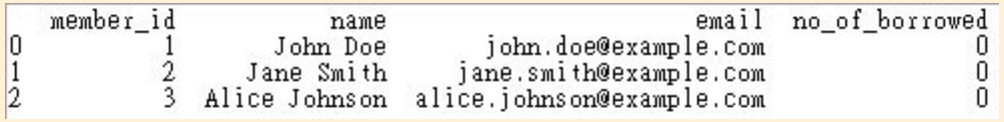
View Members

Show DataFrame with members info and number of books they borrowed

- ➢ Join two tables(Members & Borrowing_Records)
- ➢ SQL: `SELECT Members.member_id, name ,email, COALESCE(no,0) AS no_of_borrowed FROM Members LEFT JOIN (SELECT member_id, COUNT(*) AS no FROM Borrowing_Records WHERE is_returned = False group by member_id) AS count_borrow ON Members.member_id = count_borrow.member_id`



```
# view members list
def view_members():
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
#       cursor.execute('SELECT * FROM Members')
#       rows = cursor.fetchall()
#       conn.close()

#       if rows:
#           for row in rows:
#               print(f"ID: {row[0]}, Name: {row[1]}, Email: {row[2]}")
#       else:
#           print("No members found.")
    sql_query = pd.read_sql_query('SELECT Members.member_id, name ,email, COALESCE(n
    conn.close()
    df = pd.DataFrame(sql_query, columns=["member_id","name","email","no_of_borrowed"])
    if df.empty:
        print("No Member.")
        messagebox.showinfo('',"No Member")
    else:
        print(df.to_string(index=False))

        output.insert(END, f'{df}\n\n')
```

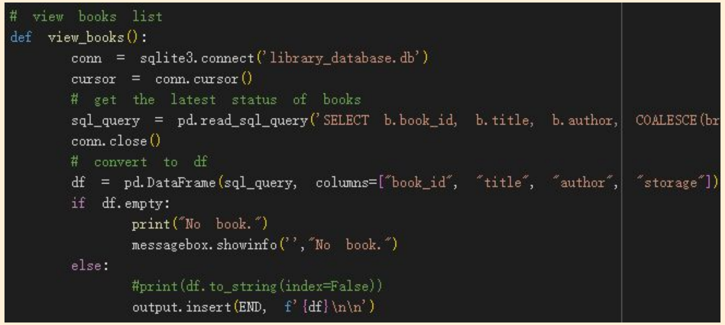


```
   member_id           name                  email  no_of_borrowed
0          1       John Doe      john.doe@example.com               0
1          2     Jane Smith    jane.smith@example.com               0
2          3  Alice Johnson  alice.johnson@example.com               0
```

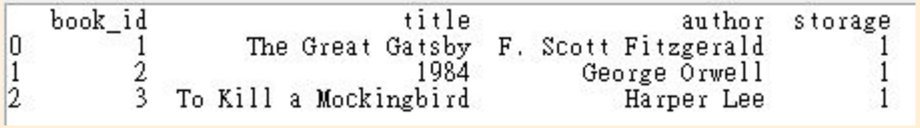
View Books

Show DataFrame with books info and storage

- ➢ Join two tables(Books & Borrowing_Records)
- ➢ SQL: `SELECT b.book_id, b.title, b.author, COALESCE(br.is_returned,True) AS storage FROM Books AS b LEFT JOIN (SELECT book_id, MAX(borrow_id) as latest_borrow_id, is_returned FROM Borrowing_records GROUP BY book_id) AS br ON b.book_id = br.book_id`

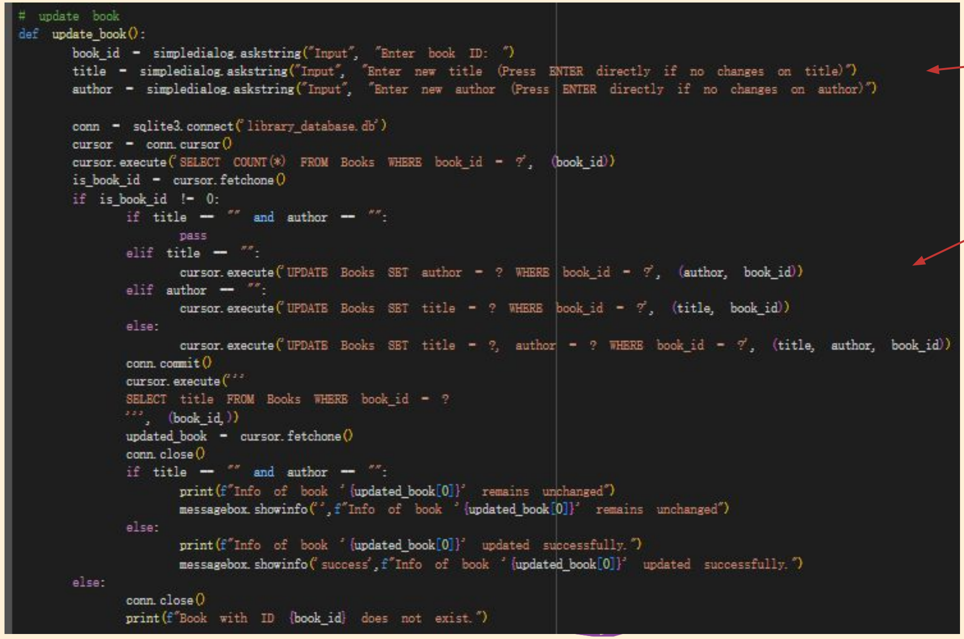


```python
# view books list
def view_books():
        conn = sqlite3.connect('library_database.db')
        cursor = conn.cursor()
        # get the latest status of books
        sql_query = pd.read_sql_query('SELECT b.book_id, b.title, b.author, COALESCE(br
        conn.close()
        # convert to df
        df = pd.DataFrame(sql_query, columns=["book_id", "title", "author", "storage"])
        if df.empty:
                print("No book.")
                messagebox.showinfo('',"No book.")
        else:
                #print(df.to_string(index=False))
                output.insert(END, f'{df}\n\n')
```



```
   book_id                 title             author  storage
0        1      The Great Gatsby  F. Scott Fitzgerald        1
1        2                  1984      George Orwell        1
2        3  To Kill a Mockingbird         Harper Lee        1
```

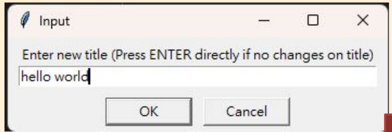
Update Books



```python
# update book
def update_book():
    book_id = simpledialog.askstring("Input", "Enter book ID: ")
    title = simpledialog.askstring("Input", "Enter new title (Press ENTER directly if no changes on title)")
    author = simpledialog.askstring("Input", "Enter new author (Press ENTER directly if no changes on author)")

    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('SELECT COUNT(*) FROM Books WHERE book_id = ?', (book_id))
    is_book_id = cursor.fetchone()
    if is_book_id != 0:
        if title == "" and author == "":
            pass
        elif title == "":
            cursor.execute('UPDATE Books SET author = ? WHERE book_id = ?', (author, book_id))
        elif author == "":
            cursor.execute('UPDATE Books SET title = ? WHERE book_id = ?', (title, book_id))
        else:
            cursor.execute('UPDATE Books SET title = ?, author = ? WHERE book_id = ?', (title, author, book_id))
        conn.commit()
        cursor.execute('''
        SELECT title FROM Books WHERE book_id = ?
        ''', (book_id,))
        updated_book = cursor.fetchone()
        conn.close()
        if title == "" and author == "":
            print(f"Info of book '{updated_book[0]}' remains unchanged")
            messagebox.showinfo('',f"Info of book '{updated_book[0]}' remains unchanged")
        else:
            print(f"Info of book '{updated_book[0]}' updated successfully.")
            messagebox.showinfo('success',f"Info of book '{updated_book[0]}' updated successfully.")
    else:
        conn.close()
        print(f"Book with ID {book_id} does not exist.")
```

Get book_id, updated title, updated author

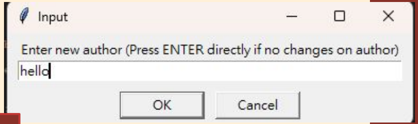
Update title, author. Title or author remain unchanged if title or author is ""



Input

Enter new title (Press ENTER directly if no changes on title)

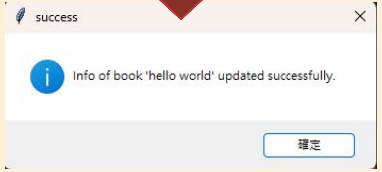
hello world



Input

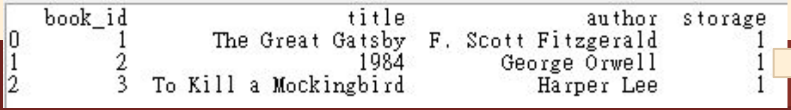
Enter new author (Press ENTER directly if no changes on author)

hello

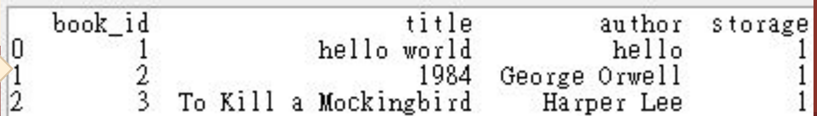


success

Info of book 'hello world' updated successfully.

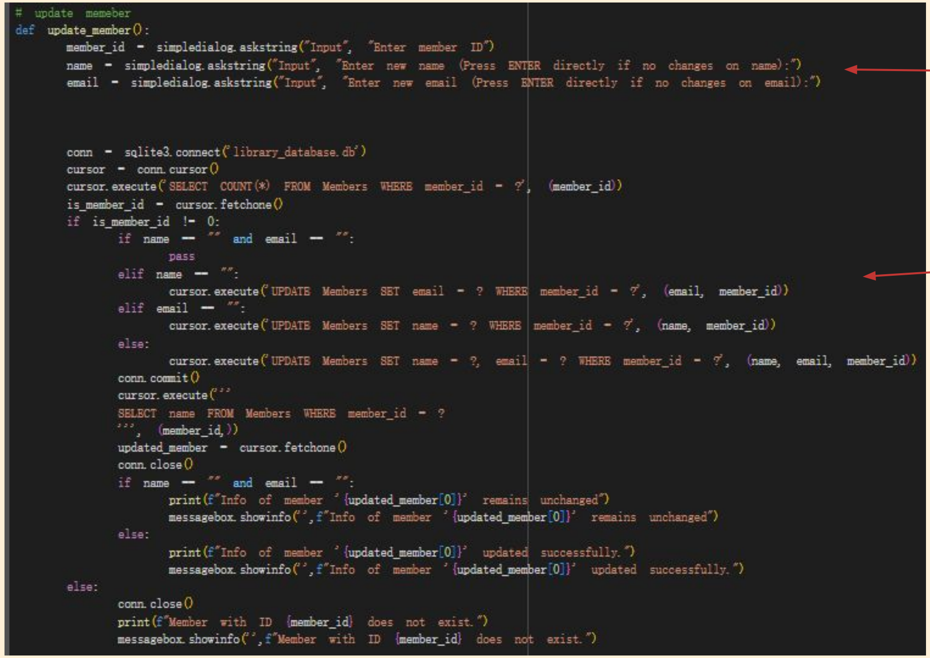


```
   book_id                 title             author  storage
0        1      The Great Gatsby  F. Scott Fitzgerald        1
1        2                  1984       George Orwell        1
2        3  To Kill a Mockingbird         Harper Lee        1
```



```
   book_id                  title         author  storage
0        1            hello world          hello        1
1        2                   1984  George Orwell        1
2        3  To Kill a Mockingbird     Harper Lee        1
```

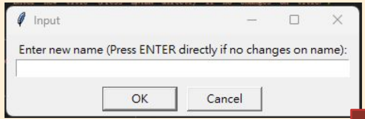
Update Members



```
# update memeber
def update_member():
    member_id = simpledialog.askstring("Input", "Enter member ID")
    name = simpledialog.askstring("Input", "Enter new name (Press ENTER directly if no changes on name):")
    email = simpledialog.askstring("Input", "Enter new email (Press ENTER directly if no changes on email):")



    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('SELECT COUNT(*) FROM Members WHERE member_id = ?', (member_id))
    is_member_id = cursor.fetchone()
    if is_member_id != 0:
        if name == "" and email == "":
            pass
        elif name == "":
            cursor.execute('UPDATE Members SET email = ? WHERE member_id = ?', (email, member_id))
        elif email == "":
            cursor.execute('UPDATE Members SET name = ? WHERE member_id = ?', (name, member_id))
        else:
            cursor.execute('UPDATE Members SET name = ?, email = ? WHERE member_id = ?', (name, email, member_id))
        conn.commit()
        cursor.execute('''
        SELECT name FROM Members WHERE member_id = ?
        ''', (member_id,))
        updated_member = cursor.fetchone()
        conn.close()
        if name == "" and email == "":
            print(f"Info of member '{updated_member[0]}' remains unchanged")
            messagebox.showinfo('',f"Info of member '{updated_member[0]}' remains unchanged")
        else:
            print(f"Info of member '{updated_member[0]}' updated successfully.")
            messagebox.showinfo('',f"Info of member '{updated_member[0]}' updated successfully.")
    else:
        conn.close()
        print(f"Member with ID {member_id} does not exist.")
        messagebox.showinfo('',f"Member with ID {member_id} does not exist.")
```

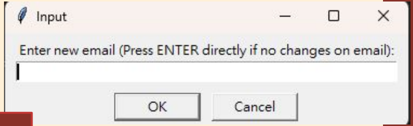
Get member_id, updated name,updated email

Update name, email. Name or email remain unchanged if name or email is ""



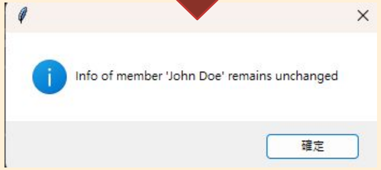
Input

Enter new name (Press ENTER directly if no changes on name):



Input

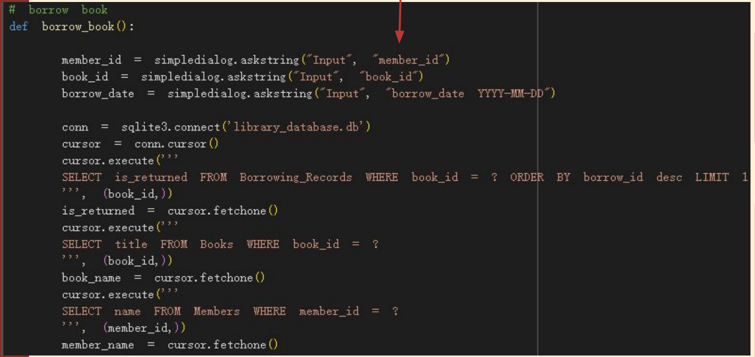
Enter new email (Press ENTER directly if no changes on email):



Info of member 'John Doe' remains unchanged

Borrow Book

Get member ID, book ID, and borrow date

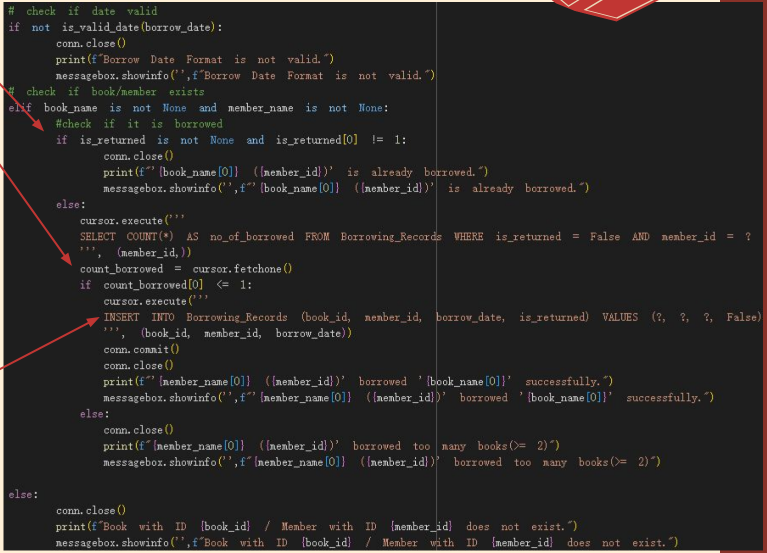


```
# borrow book
def borrow_book():

    member_id = simpledialog.askstring("Input", "member_id")
    book_id = simpledialog.askstring("Input", "book_id")
    borrow_date = simpledialog.askstring("Input", "borrow_date YYYY-MM-DD")

    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('''
    SELECT is_returned FROM Borrowing_Records WHERE book_id = ? ORDER BY borrow_id desc LIMIT 1
    ''', (book_id,))
    is_returned = cursor.fetchone()
    cursor.execute('''
    SELECT title FROM Books WHERE book_id = ?
    ''', (book_id,))
    book_name = cursor.fetchone()
    cursor.execute('''
    SELECT name FROM Members WHERE member_id = ?
    ''', (member_id,))
    member_name = cursor.fetchone()
```

Checks if the book is already borrowed

Check if member borrowed over 2 books

Add book_id, memeber_id and borrow_date to Borrowing_Records

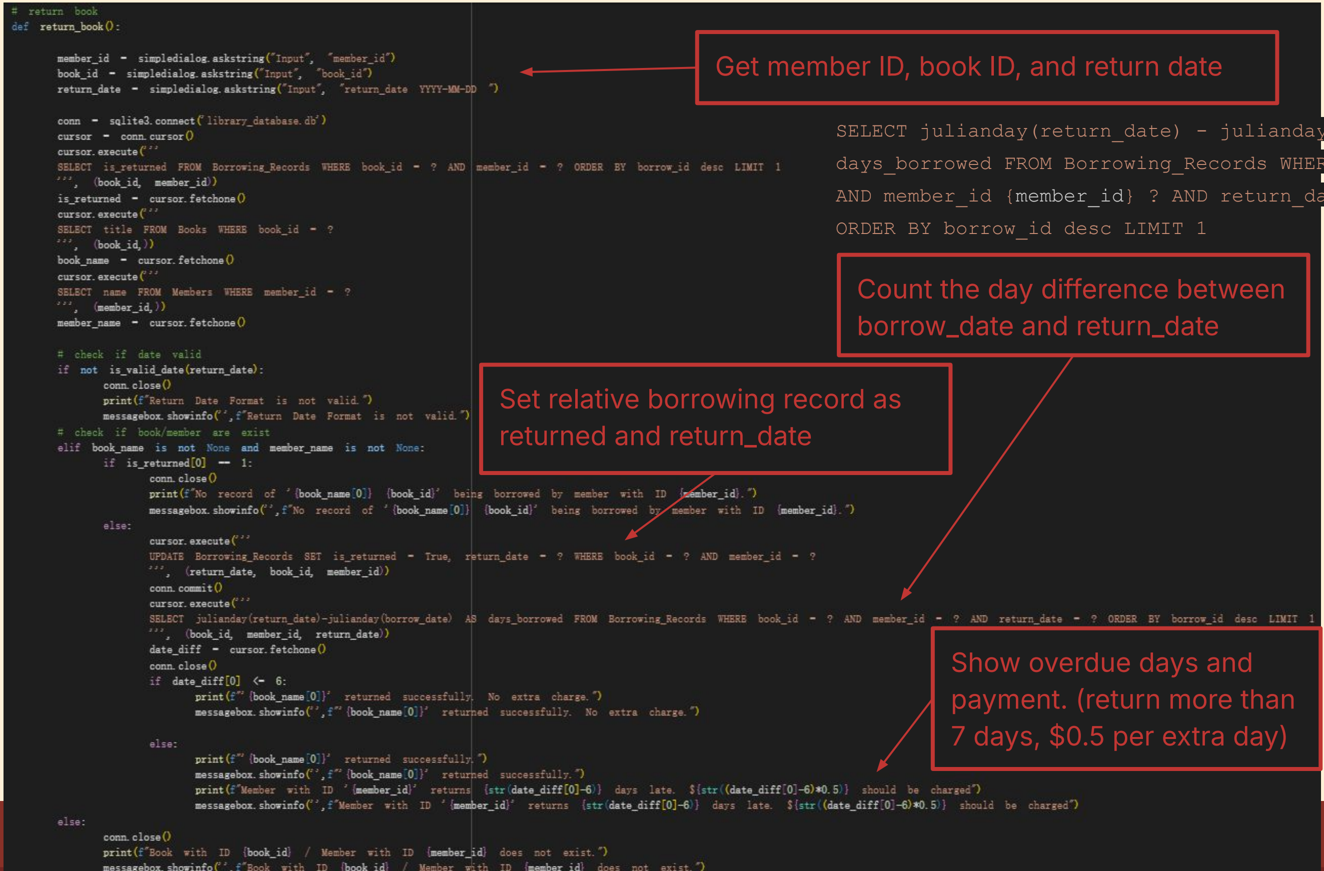


```
# check if date valid
if not is_valid_date(borrow_date):
    conn.close()
    print(f"Borrow Date Format is not valid.")
    messagebox.showinfo('',f"Borrow Date Format is not valid.")
# check if book/member exists
elif book_name is not None and member_name is not None:
    #check if it is borrowed
    if is_returned is not None and is_returned[0] != 1:
        conn.close()
        print(f"'{book_name[0]} ({member_id})' is already borrowed.")
        messagebox.showinfo('',f"'{book_name[0]} ({member_id})' is already borrowed.")
    else:
      cursor.execute('''
      SELECT COUNT(*) AS no_of_borrowed FROM Borrowing_Records WHERE is_returned = False AND member_id = ?
      ''', (member_id,))
      count_borrowed = cursor.fetchone()
      if count_borrowed[0] <= 1:
        cursor.execute('''
        INSERT INTO Borrowing_Records (book_id, member_id, borrow_date, is_returned) VALUES (?, ?, ?, False)
        ''', (book_id, member_id, borrow_date))
        conn.commit()
        conn.close()
        print(f"'{member_name[0]} ({member_id})' borrowed '{book_name[0]}' successfully.")
        messagebox.showinfo('',f"'{member_name[0]} ({member_id})' borrowed '{book_name[0]}' successfully.")
      else:
        conn.close()
        print(f"{member_name[0]} ({member_id})' borrowed too many books(>= 2)")
        messagebox.showinfo('',f"{member_name[0]} ({member_id})' borrowed too many books(>= 2)")

else:
    conn.close()
    print(f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
    messagebox.showinfo('',f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
```

Count no of borrow book:

```
SELECT COUNT(*) AS no_of_borrowed FROM
Borrowing_Records WHERE is_returned =
False AND member_id = {member_id}
```

Return Book



```python
# return book
def return_book():

    member_id = simpledialog.askstring("Input", "member_id")
    book_id = simpledialog.askstring("Input", "book_id")
    return_date = simpledialog.askstring("Input", "return_date YYYY-MM-DD ")

    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('''
    SELECT is_returned FROM Borrowing_Records WHERE book_id = ? AND member_id = ? ORDER BY borrow_id desc LIMIT 1
    ''', (book_id, member_id))
    is_returned = cursor.fetchone()
    cursor.execute('''
    SELECT title FROM Books WHERE book_id = ?
    ''', (book_id,))
    book_name = cursor.fetchone()
    cursor.execute('''
    SELECT name FROM Members WHERE member_id = ?
    ''', (member_id,))
    member_name = cursor.fetchone()

    # check if date valid
    if not is_valid_date(return_date):
        conn.close()
        print(f"Return Date Format is not valid.")
        messagebox.showinfo('',f"Return Date Format is not valid.")
    # check if book/member are exist
    elif book_name is not None and member_name is not None:
        if is_returned[0] == 1:
            conn.close()
            print(f"No record of '{book_name[0]} {book_id}' being borrowed by member with ID {member_id}.")
            messagebox.showinfo('',f"No record of '{book_name[0]} {book_id}' being borrowed by member with ID {member_id}.")
        else:
            cursor.execute('''
            UPDATE Borrowing_Records SET is_returned = True, return_date = ? WHERE book_id = ? AND member_id = ?
            ''', (return_date, book_id, member_id))
            conn.commit()
            cursor.execute('''
            SELECT julianday(return_date)-julianday(borrow_date) AS days_borrowed FROM Borrowing_Records WHERE book_id = ? AND member_id = ? AND return_date = ? ORDER BY borrow_id desc LIMIT 1
            ''', (book_id, member_id, return_date))
            date_diff = cursor.fetchone()
            conn.close()
            if date_diff[0] <= 6:
                print(f"'{book_name[0]}' returned successfully. No extra charge.")
                messagebox.showinfo('',f"'{book_name[0]}' returned successfully. No extra charge.")

            else:
                print(f"'{book_name[0]}' returned successfully.")
                messagebox.showinfo('',f"'{book_name[0]}' returned successfully.")
                print(f"Member with ID '{member_id}' returns {str(date_diff[0]-6)} days late. ${str((date_diff[0]-6)*0.5)} should be charged")
                messagebox.showinfo('',f"Member with ID '{member_id}' returns {str(date_diff[0]-6)} days late. ${str((date_diff[0]-6)*0.5)} should be charged")
    else:
        conn.close()
        print(f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
        messagebox.showinfo('',f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
```

Get member ID, book ID, and return date

```sql
SELECT julianday(return_date) - julianday(borrow_date) AS
days_borrowed FROM Borrowing_Records WHERE book_id = {book id}
AND member_id {member_id} ? AND return_date = {return date}
ORDER BY borrow_id desc LIMIT 1
```

Count the day difference between borrow_date and return_date

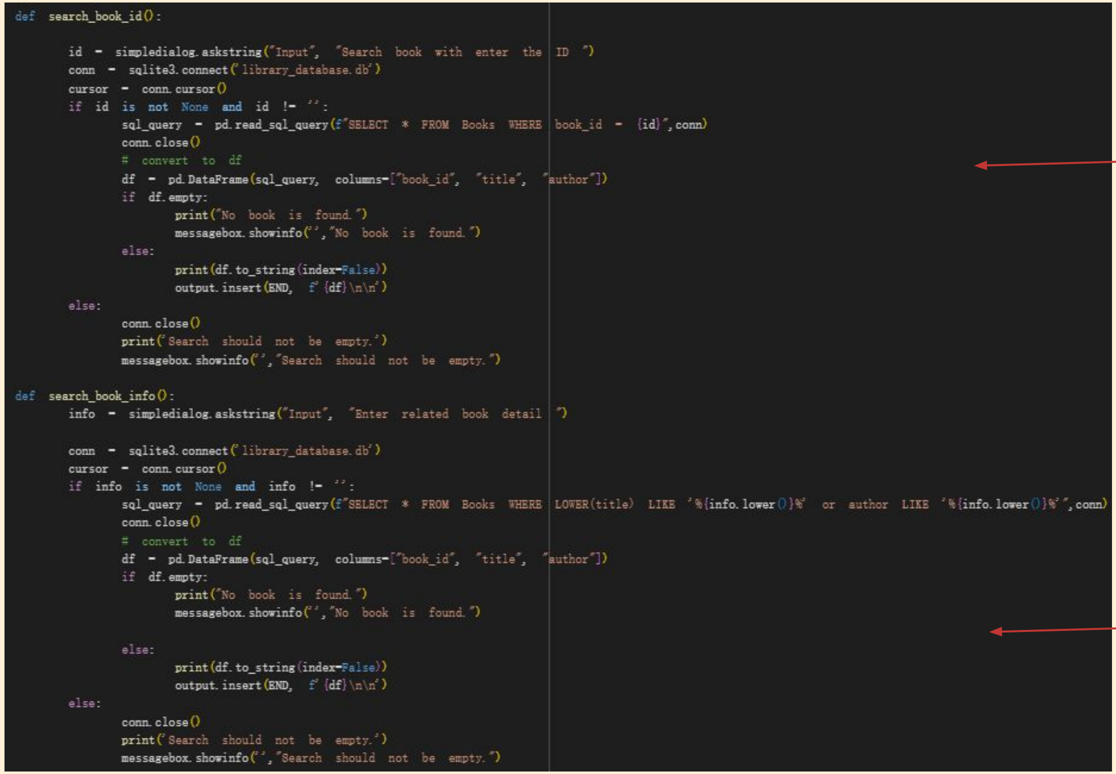
Set relative borrowing record as returned and return_date

Show overdue days and payment. (return more than 7 days, $0.5 per extra day)

03

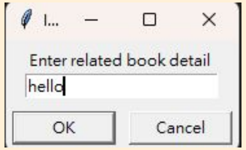
Extra Components

Search Book



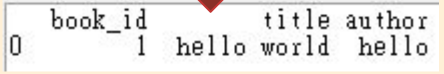
```
def search_book_id():

    id = simpledialog.askstring("Input", "Search book with enter the ID ")
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    if id is not None and id != '':
        sql_query = pd.read_sql_query(f"SELECT * FROM Books WHERE book_id = {id}",conn)
        conn.close()
        # convert to df
        df = pd.DataFrame(sql_query, columns=["book_id", "title", "author"])
        if df.empty:
            print("No book is found.")
            messagebox.showinfo('',"No book is found.")
        else:
            print(df.to_string(index=False))
            output.insert(END, f'{df}\n\n')
    else:
        conn.close()
        print('Search should not be empty.')
        messagebox.showinfo('',"Search should not be empty.")

def search_book_info():
    info = simpledialog.askstring("Input", "Enter related book detail ")

    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    if info is not None and info != '':
        sql_query = pd.read_sql_query(f"SELECT * FROM Books WHERE LOWER(title) LIKE '%{info.lower()}%' or author LIKE '%{info.lower()}%'",conn)
        conn.close()
        # convert to df
        df = pd.DataFrame(sql_query, columns=["book_id", "title", "author"])
        if df.empty:
            print("No book is found.")
            messagebox.showinfo('',"No book is found.")

        else:
            print(df.to_string(index=False))
            output.insert(END, f'{df}\n\n')
    else:
        conn.close()
        print('Search should not be empty.')
        messagebox.showinfo('',"Search should not be empty.")
```

Search books by book_id



Enter related book detail

hello

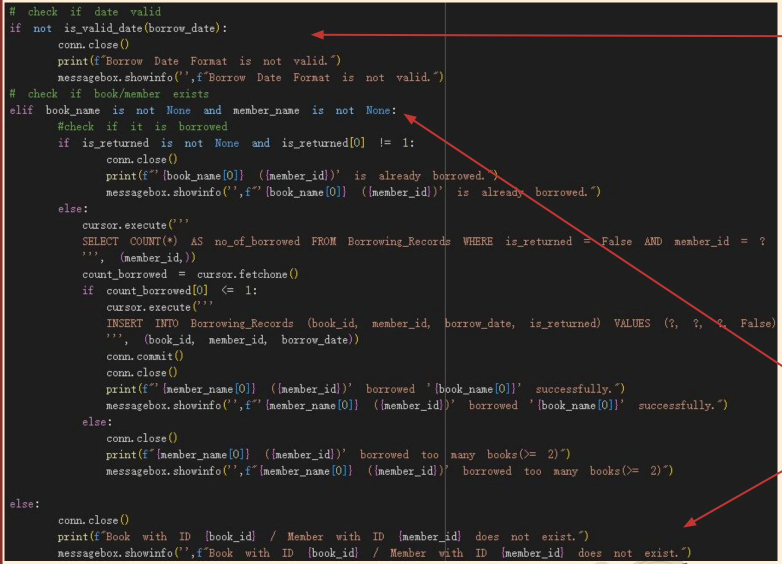


```
   book_id        title author
0        1  hello world  hello
```

Search books by book details(get all like items from title and author)

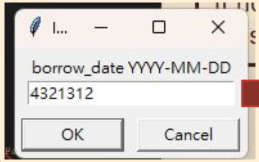
Data Validation

borrow_book() Function



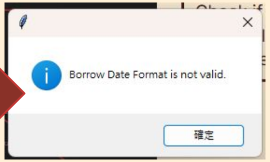
```python
# check if date valid
if not is_valid_date(borrow_date):
    conn.close()
    print(f"Borrow Date Format is not valid.")
    messagebox.showinfo('',f"Borrow Date Format is not valid.")
# check if book/member exists
elif book_name is not None and member_name is not None:
    #check if it is borrowed
    if is_returned is not None and is_returned[0] != 1:
        conn.close()
        print(f"'{book_name[0]} ({member_id})' is already borrowed.")
        messagebox.showinfo('',f"'{book_name[0]} ({member_id})' is already borrowed.")
    else:
        cursor.execute('''
        SELECT COUNT(*) AS no_of_borrowed FROM Borrowing_Records WHERE is_returned = False AND member_id = ?
        ''', (member_id,))
        count_borrowed = cursor.fetchone()
        if count_borrowed[0] <= 1:
            cursor.execute('''
            INSERT INTO Borrowing_Records (book_id, member_id, borrow_date, is_returned) VALUES (?, ?, ?, False)
            ''', (book_id, member_id, borrow_date))
            conn.commit()
            conn.close()
            print(f"'{member_name[0]} ({member_id})' borrowed '{book_name[0]}' successfully.")
            messagebox.showinfo('',f"'{member_name[0]} ({member_id})' borrowed '{book_name[0]}' successfully.")
        else:
            conn.close()
            print(f"{member_name[0]} ({member_id})' borrowed too many books(>= 2)")
            messagebox.showinfo('',f"{member_name[0]} ({member_id})' borrowed too many books(>= 2)")

else:
    conn.close()
    print(f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
    messagebox.showinfo('',f"Book with ID {book_id} / Member with ID {member_id} does not exist.")
```

Check if date format is valid. If not valid, show error message

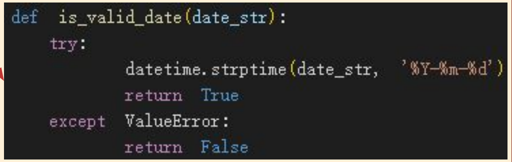


borrow_date YYYY-MM-DD

4321312



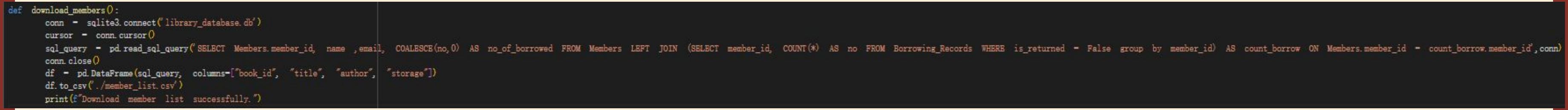
Borrow Date Format is not valid.



```python
def is_valid_date(date_str):
    try:
        datetime.strptime(date_str, '%Y-%m-%d')
        return True
    except ValueError:
        return False
```

Check if book_name and member_name are not None. If None, show error message

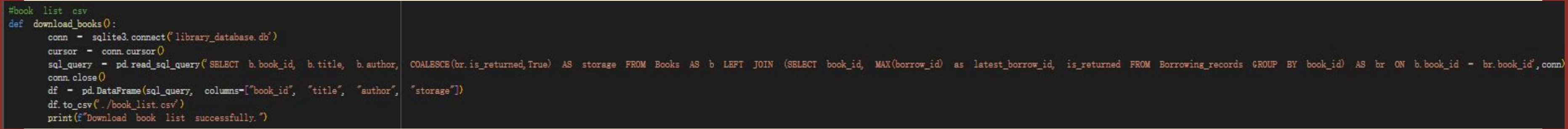
Export book/ member list to a CSV file



```python
def download_members():
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    sql_query = pd.read_sql_query('SELECT Members.member_id, name ,email, COALESCE(no,0) AS no_of_borrowed FROM Members LEFT JOIN (SELECT member_id, COUNT(*) AS no FROM Borrowing_Records WHERE is_returned = False group by member_id) AS count_borrow ON Members.member_id = count_borrow.member_id',conn)
    conn.close()
    df = pd.DataFrame(sql_query, columns=["book_id", "title", "author", "storage"])
    df.to_csv('./member_list.csv')
    print(f"Download member list successfully.")
```

Get member list. Download the list as a CSV (member_list.csv)

```sql
SELECT Members.member_id, name ,email, COALESCE(no,0)
AS no_of_borrowed FROM Members LEFT JOIN (SELECT
member_id, COUNT(*) AS no FROM Borrowing_Records WHERE
is_returned = False group by member_id) AS count_borrow
ON Members.member_id = count_borrow.member_id
```



```python
#book list csv
def download_books():
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    sql_query = pd.read_sql_query('SELECT b.book_id, b.title, b.author, COALESCE(br.is_returned,True) AS storage FROM Books AS b LEFT JOIN (SELECT book_id, MAX(borrow_id) as latest_borrow_id, is_returned FROM Borrowing_records GROUP BY book_id) AS br ON b.book_id = br.book_id',conn)
    conn.close()
    df = pd.DataFrame(sql_query, columns=["book_id", "title", "author", "storage"])
    df.to_csv('./book_list.csv')
    print(f"Download book list successfully.")
```

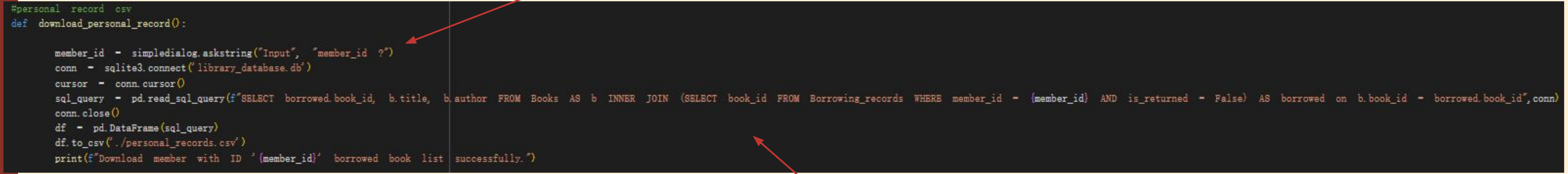
Get book list. Download the list as a CSV (book_list.csv)

```sql
SELECT b.book_id, b.title, b.author,
COALESCE(br.is_returned,True) AS storage FROM Books
AS b LEFT JOIN (SELECT book_id, MAX(borrow_id) as
latest_borrow_id, is_returned FROM Borrowing_records
GROUP BY book_id) AS br ON b.book_id = br.book_id
```

Export personal list to a CSV file

```
SELECT borrowed.book_id, b.title, b.author
FROM Books AS b INNER JOIN (SELECT book_id
FROM Borrowing_records WHERE member_id =
{member_id} AND is_returned = False) AS
borrowed on b.book_id = borrowed.book_id
```

Get member_id



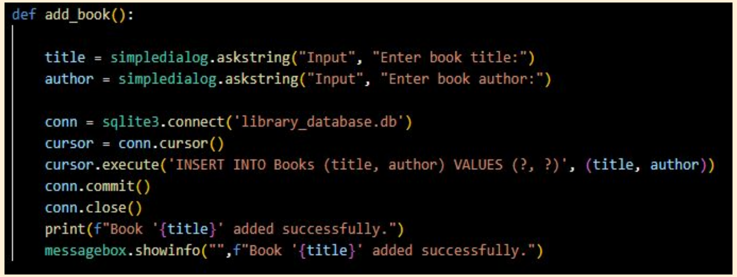
```
#personal record csv
def download_personal_record():

        member_id = simpledialog.askstring("Input", "member_id ?")
        conn = sqlite3.connect('library_database.db')
        cursor = conn.cursor()
        sql_query = pd.read_sql_query(f"SELECT borrowed.book_id, b.title, b.author FROM Books AS b INNER JOIN (SELECT book_id FROM Borrowing_records WHERE member_id = {member_id} AND is_returned = False) AS borrowed on b.book_id = borrowed.book_id",conn)
        conn.close()
        df = pd.DataFrame(sql_query)
        df.to_csv('./personal_records.csv')
        print(f"Download member with ID '{member_id}' borrowed book list successfully.")
```

Get personal borrow book list. Download the list as a CSV (personal_records.csv)

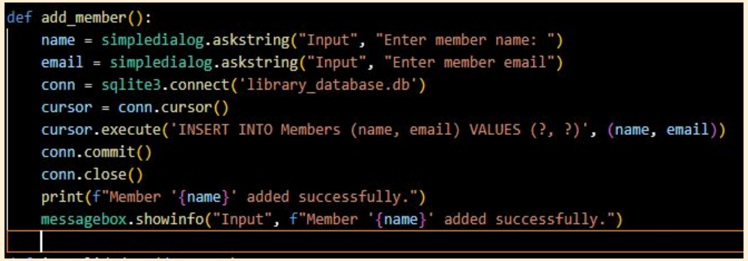
```
INSERT INTO Books (title, author)
VALUES (?, ?)', (title, author)
```

inserts into an SQLite database

```
INSERT INTO Members (name, email)
VALUES (?, ?)', (name, email)
```



```python
def add_book():

    title = simpledialog.askstring("Input", "Enter book title:")
    author = simpledialog.askstring("Input", "Enter book author:")

    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO Books (title, author) VALUES (?, ?)', (title, author))
    conn.commit()
    conn.close()
    print(f"Book '{title}' added successfully.")
    messagebox.showinfo("",f"Book '{title}' added successfully.")
```

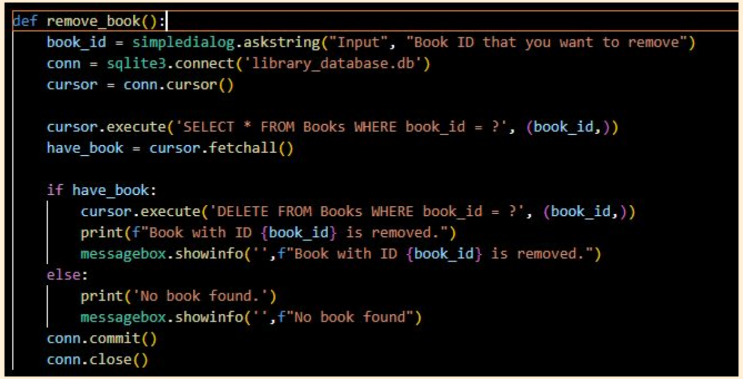


```python
def add_member():
    name = simpledialog.askstring("Input", "Enter member name: ")
    email = simpledialog.askstring("Input", "Enter member email")
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO Members (name, email) VALUES (?, ?)', (name, email))
    conn.commit()
    conn.close()
    print(f"Member '{name}' added successfully.")
    messagebox.showinfo("Input", f"Member '{name}' added successfully.")
```

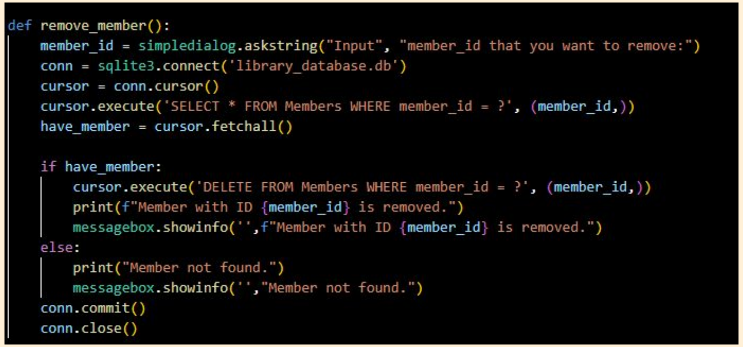
```
DELETE FROM Books WHERE book_id =
?', (book_id,)
```

handle user input to remove books and members from the database

```
DELETE FROM Members WHERE member_id
= ?', (member_id,)
```



```python
def remove_book():
    book_id = simpledialog.askstring("Input", "Book ID that you want to remove")
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()

    cursor.execute('SELECT * FROM Books WHERE book_id = ?', (book_id,))
    have_book = cursor.fetchall()

    if have_book:
        cursor.execute('DELETE FROM Books WHERE book_id = ?', (book_id,))
        print(f"Book with ID {book_id} is removed.")
        messagebox.showinfo('',f"Book with ID {book_id} is removed.")
    else:
        print('No book found.')
        messagebox.showinfo('',f"No book found")
    conn.commit()
    conn.close()
```



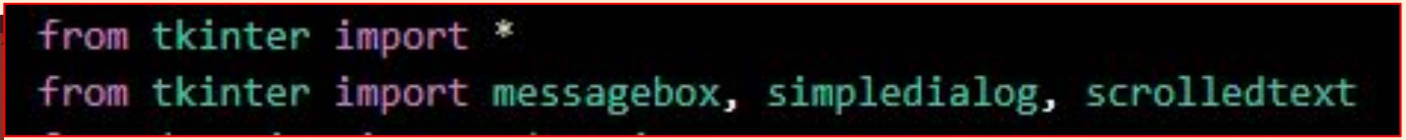
```python
def remove_member():
    member_id = simpledialog.askstring("Input", "member_id that you want to remove:")
    conn = sqlite3.connect('library_database.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM Members WHERE member_id = ?', (member_id,))
    have_member = cursor.fetchall()

    if have_member:
        cursor.execute('DELETE FROM Members WHERE member_id = ?', (member_id,))
        print(f"Member with ID {member_id} is removed.")
        messagebox.showinfo('',f"Member with ID {member_id} is removed.")
    else:
        print("Member not found.")
        messagebox.showinfo('',"Member not found.")
    conn.commit()
    conn.close()
```

Front-END

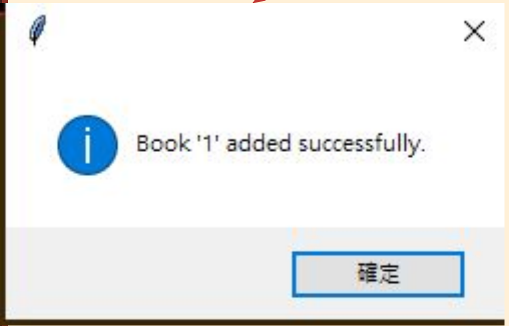
03

GUI tkinter

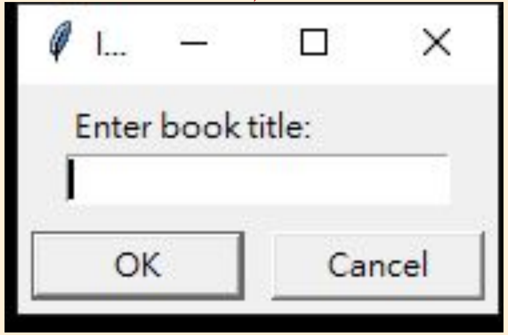
essential component



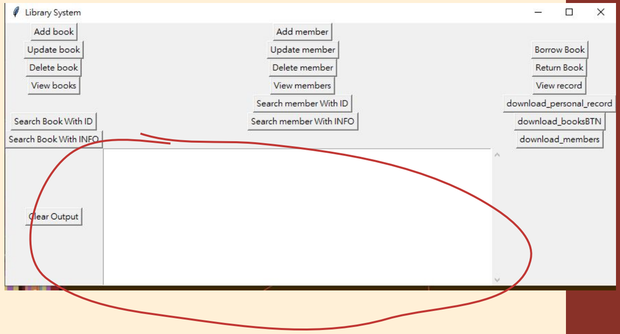
```
from tkinter import *
from tkinter import messagebox, simpledialog, scrolledtext
```

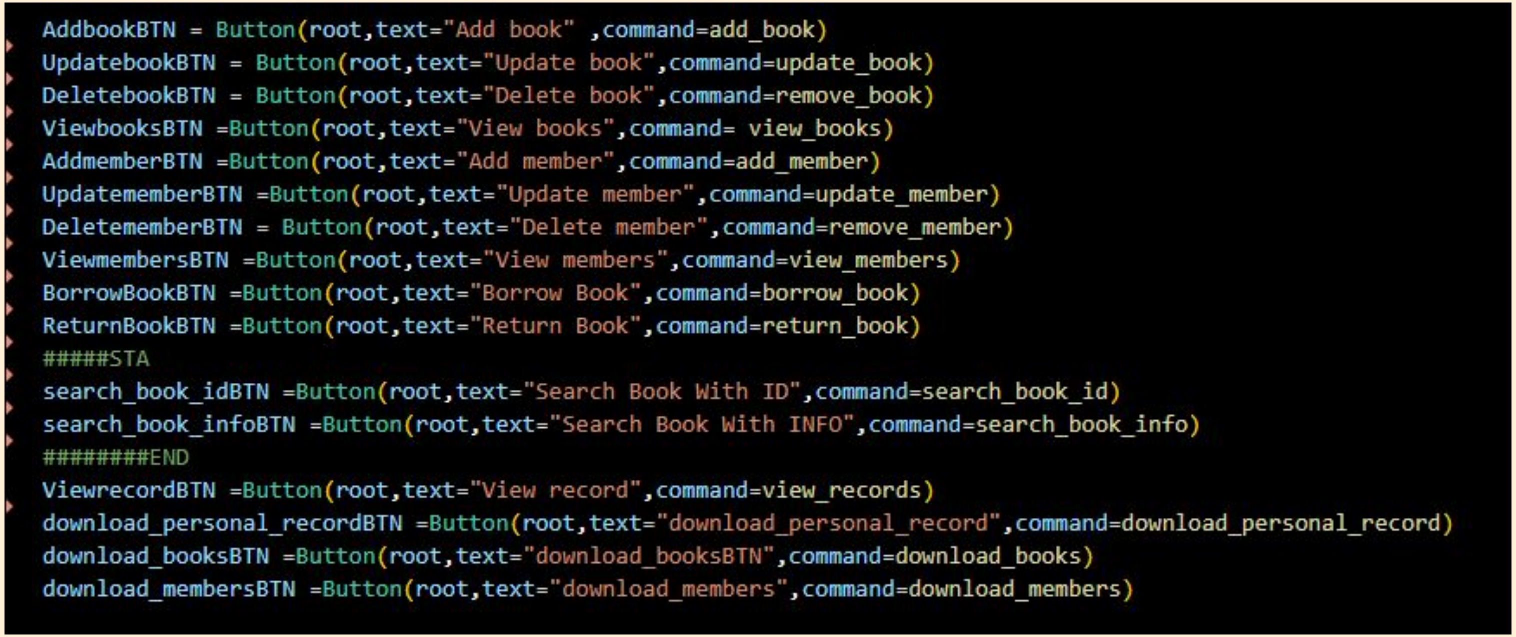


Book '1' added successfully.

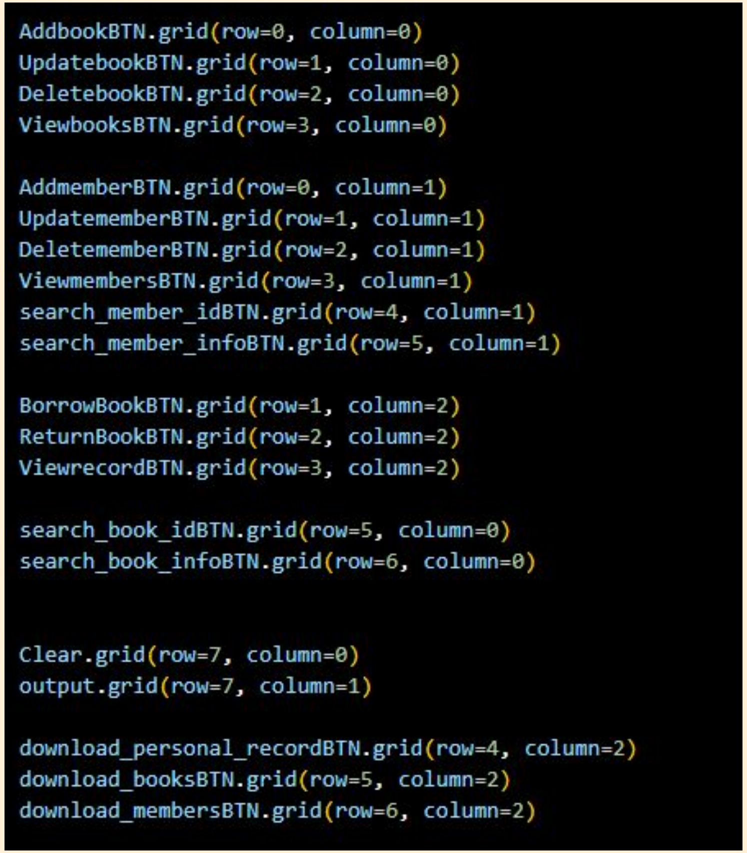


Enter book title:

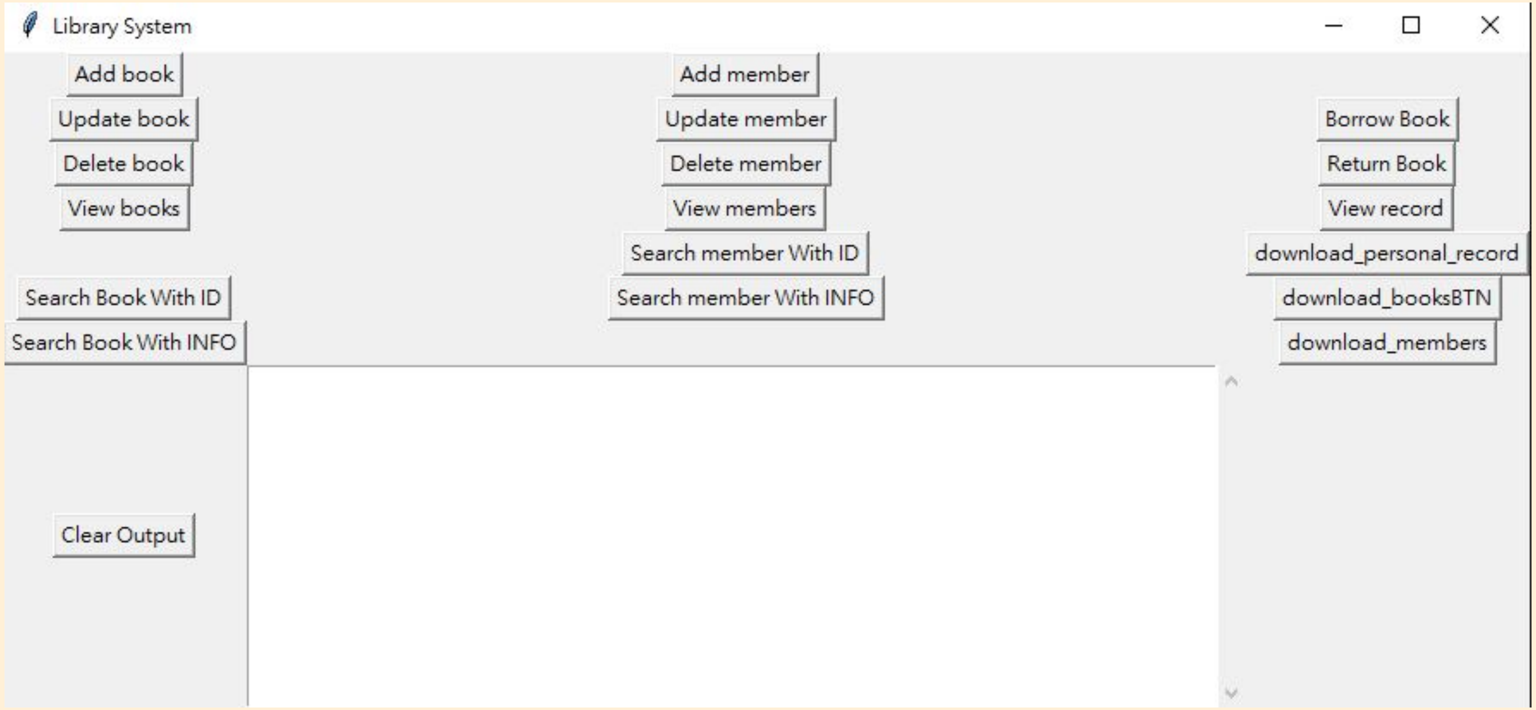




```
AddbookBTN = Button(root,text="Add book"  ,command=add_book)
UpdatebookBTN = Button(root,text="Update book",command=update_book)
DeletebookBTN = Button(root,text="Delete book",command=remove_book)
ViewbooksBTN =Button(root,text="View books",command= view_books)
AddmemberBTN =Button(root,text="Add member",command=add_member)
UpdatememberBTN =Button(root,text="Update member",command=update_member)
DeletememberBTN = Button(root,text="Delete member",command=remove_member)
ViewmembersBTN =Button(root,text="View members",command=view_members)
BorrowBookBTN =Button(root,text="Borrow Book",command=borrow_book)
ReturnBookBTN =Button(root,text="Return Book",command=return_book)
#####STA
search_book_idBTN =Button(root,text="Search Book With ID",command=search_book_id)
search_book_infoBTN =Button(root,text="Search Book With INFO",command=search_book_info)
########END
ViewrecordBTN =Button(root,text="View record",command=view_records)
download_personal_recordBTN =Button(root,text="download_personal_record",command=download_personal_record)
download_booksBTN =Button(root,text="download_booksBTN",command=download_books)
download_membersBTN =Button(root,text="download_members",command=download_members)
```



```
AddbookBTN.grid(row=0, column=0)
UpdatebookBTN.grid(row=1, column=0)
DeletebookBTN.grid(row=2, column=0)
ViewbooksBTN.grid(row=3, column=0)

AddmemberBTN.grid(row=0, column=1)
UpdatememberBTN.grid(row=1, column=1)
DeletememberBTN.grid(row=2, column=1)
ViewmembersBTN.grid(row=3, column=1)
search_member_idBTN.grid(row=4, column=1)
search_member_infoBTN.grid(row=5, column=1)

BorrowBookBTN.grid(row=1, column=2)
ReturnBookBTN.grid(row=2, column=2)
ViewrecordBTN.grid(row=3, column=2)

search_book_idBTN.grid(row=5, column=0)
search_book_infoBTN.grid(row=6, column=0)


Clear.grid(row=7, column=0)
output.grid(row=7, column=1)

download_personal_recordBTN.grid(row=4, column=2)
download_booksBTN.grid(row=5, column=2)
download_membersBTN.grid(row=6, column=2)
```


Library System
Add book
Update book
Delete book
View books
Search Book With ID
Search Book With INFO
Clear Output
Add member
Update member
Delete member
View members
Search member With ID
Search member With INFO
Borrow Book
Return Book
View record
download_personal_record
download_booksBTN
download_members

04

Demo

You can enter a subtitle here if you need it

DEMO TIME

END~