```
# Importing all necessary modules
import sqlite3

from tkinter import *
import tkinter.ttk as ttk
import tkinter.messagebox as mb
import tkinter.simpledialog as sd

# Connecting to Database
connector = sqlite3.connect('library.db')
cursor = connector.cursor()

connector.execute(
'CREATE TABLE IF NOT EXISTS Library (BK_NAME TEXT, BK_ID TEXT PRIMARY KEY NOT NULL, AUTHOR_NAME TEXT, BK_STATUS TEXT, CARD_ID TEXT)'
)
```

```
def issuer_card():
    Cid = sd.askstring('Issuer Card ID', 'What is the Issuer\'s Card ID?\t\t\t')

    if not Cid:
     mb.showerror('Issuer ID cannot be zero!', 'Can\'t keep Issuer ID empty, it must have a value')
    else:
      return Cid


def display_records():
    global connector, cursor
    global tree

    tree.delete(*tree.get_children())

    curr = connector.execute('SELECT * FROM Library')
    data = curr.fetchall()

    for records in data:
        tree.insert('', END, values=records)


def clear_fields():
    global bk_status, bk_id, bk_name, author_name, card_id

    bk_status.set('Available')
    for i in ['bk_id', 'bk_name', 'author_name', 'card_id']:
     exec(f"{i}.set('')")
     bk_id_entry.config(state='normal')
    try:
      tree.selection_remove(tree.selection()[0])
    except:
           pass


def clear_and_display():
     clear_fields()
     display_records()


def view_record():
         global bk_name, bk_id, bk_status, author_name, card_id
         global tree

         if not tree.focus():
           mb.showerror('Select a row!', 'To view a record, you must select it in the table. Please do so before continuing.')
           return

         current_item_selected = tree.focus()
         values_in_selected_item = tree.item(current_item_selected)
         selection = values_in_selected_item['values']

         bk_name.set(selection[0]) ; bk_id.set(selection[1]) ; bk_status.set(selection[3])
         author_name.set(selection[2])
```

```
def add_record():
    global connector
    global bk_name, bk_id, author_name, bk_status

    if bk_status.get() == 'Issued':
    card_id.set(issuer_card())
    else:
    card_id.set('N/A')

    surety = mb.askyesno('Are you sure?',
                  'Are you sure this is the data you want to enter?\nPlease note that Book ID cannot be changed in the future')

    if surety:
          try:
                connector.execute(
                 'INSERT INTO Library (BK_NAME, BK_ID, AUTHOR_NAME, BK_STATUS, CARD_ID) VALUES (?, ?, ?, ?, ?)',
                        (bk_name.get(), bk_id.get(), author_name.get(), bk_status.get(), card_id.get()))
                connector.commit()

                clear_and_display()

                mb.showinfo('Record added', 'The new record was successfully added to your database')
      except sqlite3.IntegrityError:
                mb.showerror('Book ID already in use!',
                             'The Book ID you are trying to enter is already in the database, please alter that book\'s record or check any discrepancies on your side')


def update_record():
   def update():
       global bk_status, bk_name, bk_id, author_name, card_id
       global connector, tree

       if bk_status.get() == 'Issued':
             card_id.set(issuer_card())
       else:
             card_id.set('N/A')

       cursor.execute('UPDATE Library SET BK_NAME=?, BK_STATUS=?, AUTHOR_NAME=?, CARD_ID=? WHERE BK_ID=?', (bk_name.get(), bk_status.get(), author_name.get(), card_id.get(), bk_id.get()))
        connector.commit()

        clear_and_display()

        edit.destroy()
        bk_id_entry.config(state='normal')
        clear.config(state='normal')

 view_record()

 bk_id_entry.config(state='disable')
 clear.config(state='disable')

 edit = Button(left_frame, text='Update Record', font=btn_font, bg=btn_hlb_bg, width=20, command=update)
 edit.place(x=50, y=375)


def remove_record():
 if not tree.selection():
  mb.showerror('Error!', 'Please select an item from the database')
   return

 current_item = tree.focus()
 values = tree.item(current_item)
 selection = values["values"]

 cursor.execute('DELETE FROM Library WHERE BK_ID=?', (selection[1], ))
 connector.commit()

 tree.delete(current_item)

 mb.showinfo('Done', 'The record you wanted deleted was successfully deleted.')

 clear_and_display()


def delete_inventory():
 if mb.askyesno('Are you sure?', 'Are you sure you want to delete the entire inventory?\n\n\nThis command cannot be reversed'):
  tree.delete(*tree.get_children())

  cursor.execute('DELETE FROM Library')
  connector.commit()
else:
  return


def change_availability():
 global card_id, tree, connector

 if not tree.selection():
  mb.showerror('Error!', 'Please select a book from the database')
  return

 current_item = tree.focus()
 values = tree.item(current_item)
 BK_id = values['values'][1]
 BK_status = values["values"][3]

 if BK_status == 'Issued':
  surety = mb.askyesno('Is return confirmed?', 'Has the book been returned to you?')
    if surety:
       cursor.execute('UPDATE Library SET bk_status=?, card_id=? WHERE bk_id=?', ('Available', 'N/A', BK_id))
       connector.commit()
 else: mb.showinfo(
      'Cannot be returned', 'The book status cannot be set to Available unless it has been returned')
else:
     cursor.execute('UPDATE Library SET bk_status=?, card_id=? where bk_id=?', ('Issued', issuer_card(), BK_id))
     connector.commit()

 clear_and_display()
```

```
# Variables
lf_bg = 'LightSkyBlue' # Left Frame Background Color
rtf_bg = 'DeepSkyBlue' # Right Top Frame Background Color
rbf_bg = 'DodgerBlue' # Right Bottom Frame Background Color
btn_hlb_bg = 'SteelBlue' # Background color for Head Labels and Buttons

lbl_font = ('Georgia', 13) # Font for all labels
entry_font = ('Times New Roman', 12) # Font for all Entry widgets
btn_font = ('Gill Sans MT', 13)

# Initializing the main GUI window
root = Tk()
root.title('PythonGeeks Library Management System')
root.geometry('1010x530')
root.resizable(0, 0)

Label(root, text='LIBRARY MANAGEMENT SYSTEM', font=("Noto Sans CJK TC", 15, 'bold'), bg=btn_hlb_bg, fg='White').pack(side=TOP, fill=X)

# StringVars
bk_status = StringVar()
bk_name = StringVar()
bk_id = StringVar()
author_name = StringVar()
card_id = StringVar()

# Frames
left_frame = Frame(root, bg=lf_bg)
left_frame.place(x=0, y=30, relwidth=0.3, relheight=0.96)

RT_frame = Frame(root, bg=rtf_bg)
RT_frame.place(relx=0.3, y=30, relheight=0.2, relwidth=0.7)

RB_frame = Frame(root)
RB_frame.place(relx=0.3, rely=0.24, relheight=0.785, relwidth=0.7)

# Left Frame
Label(left_frame, text='Book Name', bg=lf_bg, font=lbl_font).place(x=98, y=25)
Entry(left_frame, width=25, font=entry_font, text=bk_name).place(x=45, y=55)

Label(left_frame, text='Book ID', bg=lf_bg, font=lbl_font).place(x=110, y=105)
bk_id_entry = Entry(left_frame, width=25, font=entry_font, text=bk_id)
bk_id_entry.place(x=45, y=135)

Label(left_frame, text='Author Name', bg=lf_bg, font=lbl_font).place(x=90, y=185)
Entry(left_frame, width=25, font=entry_font, text=author_name).place(x=45, y=215)

Label(left_frame, text='Status of the Book', bg=lf_bg, font=lbl_font).place(x=75, y=265)
dd = OptionMenu(left_frame, bk_status, *['Available', 'Issued'])
dd.configure(font=entry_font, width=12)
dd.place(x=75, y=300)

submit = Button(left_frame, text='Add new record', font=btn_font, bg=btn_hlb_bg, width=20, command=add_record)
submit.place(x=50, y=375)

clear = Button(left_frame, text='Clear fields', font=btn_font, bg=btn_hlb_bg, width=20, command=clear_fields)
clear.place(x=50, y=435)

# Right Top Frame
Button(RT_frame, text='Delete book record', font=btn_font, bg=btn_hlb_bg, width=17, command=remove_record).place(x=8, y=30)
Button(RT_frame, text='Delete full inventory', font=btn_font, bg=btn_hlb_bg, width=17, command=delete_inventory).place(x=178, y=30)
Button(RT_frame, text='Update book details', font=btn_font, bg=btn_hlb_bg, width=17,
command=update_record).place(x=348, y=30)
Button(RT_frame, text='Change Book Availability', font=btn_font, bg=btn_hlb_bg, width=19,
command=change_availability).place(x=518, y=30)

# Right Bottom Frame
Label(RB_frame, text='BOOK INVENTORY', bg=rbf_bg, font=("Noto Sans CJK TC", 15, 'bold')).pack(side=TOP, fill=X)

tree = ttk.Treeview(RB_frame, selectmode=BROWSE, columns=('Book Name', 'Book ID', 'Author', 'Status', 'Issuer Card ID'))

XScrollbar = Scrollbar(tree, orient=HORIZONTAL, command=tree.xview)
YScrollbar = Scrollbar(tree, orient=VERTICAL, command=tree.yview)
XScrollbar.pack(side=BOTTOM, fill=X)
YScrollbar.pack(side=RIGHT, fill=Y)

tree.config(xscrollcommand=XScrollbar.set, yscrollcommand=YScrollbar.set)

tree.heading('Book Name', text='Book Name', anchor=CENTER)
tree.heading('Book ID', text='Book ID', anchor=CENTER)
tree.heading('Author', text='Author', anchor=CENTER)
tree.heading('Status', text='Status of the Book', anchor=CENTER)
tree.heading('Issuer Card ID', text='Card ID of the Issuer', anchor=CENTER)

tree.column('#0', width=0, stretch=NO)
tree.column('#1', width=225, stretch=NO)
tree.column('#2', width=70, stretch=NO)
tree.column('#3', width=150, stretch=NO)
tree.column('#4', width=105, stretch=NO)
tree.column('#5', width=132, stretch=NO)

tree.place(y=30, x=0, relheight=0.9, relwidth=1)

clear_and_display()

# Finalizing the window
root.update()
root.mainloop()
```