**A. Core Python & Data Handling**

**1. Modify the project so that all data (books, members) is auto-saved every time an operation happens, without requiring manual save().**  
Call self.save() at the end of every method like add\_book, remove\_book, register\_member, etc.

def add\_book(self, book: Book):

self.books[book.book\_id] = book

self.save() # auto-save after adding

**2. Add a feature to search books by partial title or author name (case-insensitive).**  
Use .lower() to ignore case while searching.

def search\_books(self, keyword: str):

keyword = keyword.lower()

return [book for book in self.books.values()

if keyword in book.title.lower() or keyword in book.author.lower()]

**3. Implement sorting of books by title, author, or availability using Python’s sorted() and custom key functions.**

def sort\_books(self, by: str = "title"):

if by == "title":

return sorted(self.books.values(), key=lambda b: b.title)

elif by == "author":

return sorted(self.books.values(), key=lambda b: b.author)

elif by == "availability":

return sorted(self.books.values(), key=lambda b: b.available, reverse=True)

**4. Use list comprehensions to fetch all currently borrowed books.**

borrowed\_books = [book for book in self.books.values() if not book.available]

**5. Add a feature to export all library data to a CSV file.**

import csv

def export\_to\_csv(self, filename="library\_data.csv"):

with open(filename, "w", newline="") as f:

writer = csv.writer(f)

writer.writerow(["Book ID", "Title", "Author", "ISBN", "Available"])

for book in self.books.values():

writer.writerow([book.book\_id, book.title, book.author, book.isbn, book.available])

**6. Convert the book and member collections into dictionaries of dataclasses instead of normal classes.**

from dataclasses import dataclass

@dataclass

class Book:

book\_id: str

title: str

author: str

isbn: str

available: bool = True

**7. Use zip() to pair members with the books they borrowed for a custom report.**

def borrowed\_report(self):

for member in self.members.values():

paired = list(zip([member.name]\*len(member.borrowed), member.borrowed))

print(paired)

**8. Write a function that uses regular expressions to validate ISBN numbers.**

import re

def validate\_isbn(isbn: str) -> bool:

pattern = r"^\d{3}-\d{10}$" # Example: 978-1234567890

return bool(re.match(pattern, isbn))

**B. Advanced OOP Concepts**

**9. Introduce a StaffMember subclass with permission to remove books, while normal members cannot.**

class StaffMember(Member):

def remove\_book(self, library, book\_id: str):

if book\_id in library.books:

del library.books[book\_id]

library.save()

**10. Implement operator overloading (eq, lt) so two books can be compared by ISBN.**

class Book:

# existing code ...

def \_\_eq\_\_(self, other):

return self.isbn == other.isbn

def \_\_lt\_\_(self, other):

return self.isbn < other.isbn