```python
import re
import random
class Member:
    def __init__(self, name, email):
        if not self.validate_email(email):
            raise ValueError("Invalid email format")
        self.name = name
        self.email = email
        self.member_id = self.generate_member_id()
    @staticmethod
    def validate_email(email):
        """Validates the format of an email."""
        email_regex = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
        return re.match(email_regex, email) is not None
    @staticmethod
    def generate_member_id():
        """Generates a unique member ID in the format LIB1234."""
        return f"LIB{random.randint(1000, 9999)}"
    @staticmethod
    def verify_member_id(member_id):
        """Verifies the format of a member ID."""
        return bool(re.match(r'^LIB\d{4}$', member_id))
class Library(Member):
    def __init__(self):
        self.books = {}
        self.members = {}
        self.borrowed_books = {}
    def add_book(self, book_id, title, author):
        if book_id in self.books:
            print(f"Book with ID {book_id} already exists.")
            return
        self.books[book_id] = {
            'title': title,
            'author': author,
            'available': True
        }
        print(f"Book '{title}' added successfully.")
    def register_member(self, name, email):
        try:
            new_member = Member(name, email)
            if new_member.member_id in self.members:
                print(f"Member with ID {new_member.member_id} already
exists.")
                return
            self.members[new_member.member_id] = new_member
            print(f"Member '{name}' registered successfully with ID
{new_member.member_id}.")
        except ValueError as e:
            print(e)
    def borrow_book(self, member_id, book_id):
        if member_id not in self.members:
            print("Invalid member ID.")
            return
        if book_id not in self.books:
            print("Invalid book ID.")
            return
        if not self.books[book_id]['available']:
            print(f"The book '{self.books[book_id]['title']}' is currently
unavailable.")
            return
        self.books[book_id]['available'] = False
        if member_id not in self.borrowed_books:
            self.borrowed_books[member_id] = []
        self.borrowed_books[member_id].append(book_id)
        print(f"Book '{self.books[book_id]['title']}' borrowed successfully by
member {member_id}.")
    def return_book(self, member_id, book_id):
        if member_id not in self.borrowed_books or book_id not in
self.borrowed_books[member_id]:
            print("This book was not borrowed by the member.")
            return
        self.borrowed_books[member_id].remove(book_id)
        if not self.borrowed_books[member_id]:
            del self.borrowed_books[member_id]
        self.books[book_id]['available'] = True
        print(f"Book '{self.books[book_id]['title']}' returned successfully by
member {member_id}.")
library = Library()
library.add_book("B001", "1984", "George Orwell")
library.add_book("B002", "To Kill a Mockingbird", "Harper Lee")
library.register_member("Alice", "alice@gmail.com")
library.register_member("Bob", "bob@gmail.com")
member_ids = list(library.members.keys())
library.borrow_book(member_ids[0], "B001")
library.return_book(member_ids[0], "B001")
```