Design Rationale

In designing this Library Management System, we selected Python's built-in data structures—dictionaries, lists, and tuples—for their simplicity, efficiency, and alignment with the assignment's constraints, avoiding external libraries for an intermediate-level implementation.

Dictionaries for Books: A dictionary with ISBN as the key provides O(1) average-time complexity for lookups, additions, updates, and deletions—critical for unique book identification. Each value is a nested dictionary holding mutable attributes (title, author, genre, total_copies, available_copies), allowing easy CRUD operations. This structure naturally enforces ISBN uniqueness via key presence checks and supports tracking copy availability independently of member borrows, preventing over-borrowing without complex queries.

Lists for Members: A list of dictionaries suits members well, as there's no inherent ordering or frequent indexing by ID (searches are linear but acceptable for small-scale systems). Each member's dictionary includes a 'borrowed' list of ISBN strings, enabling quick append/pop for borrows/returns and easy checks for deletion eligibility (non-empty list blocks deletes). Linear iteration (O(n)) for ID lookups is fine for low member counts; for scalability, a dictionary could be added later, but lists keep it straightforward.

Tuples for Genres: Genres are a fixed, immutable set of valid options (e.g., ('Fiction', 'Non-Fiction', 'Sci-Fi')). A tuple ensures constancy—no accidental modifications—and supports efficient 'in' membership tests (O(1) average). This prevents invalid data entry during book addition, enhancing data integrity without over-engineering (e.g., no enum class needed).

These choices promote modularity: functions are pure (take/return data, no side effects beyond params), testable via assertions, and composable in demo.py. Borrow/return logic integrates seamlessly—decrementing book availability while appending to member lists—balancing normalization (no redundant copy tracking) with performance. Overall, this design is extensible (e.g., add logging) while meeting requirements with clean, readable code.