**1.Table Student**

1. Student UID (rfid UID)
2. Student first name
3. Student last name
4. CourseID (FK) – create a separate table of courses with coursed
5. Rollno
6. Email-id
7. Phone Number
8. Date of birth
9. Address
10. Issued book count – create a trigger which adds +1 whenever a student borrows book and record in inserted in borrowed\_books table.
11. Date of registration
12. End of registration
13. Status – enum Active, Inactive.

**2.Table Staff**

1. Staff UID (rfid UID)
2. Staff first name
3. Staff last name
4. Branch (FK)
5. Faculty (FK)
6. Phone number
7. Issued book count – create a trigger which adds +1 whenever a student borrows book and record in inserted in borrowed\_books table.
8. Date of registration
9. Status – enum Active, Inactive.

**3.Table Books**

1. Book UID (rfid UID)
2. Title
3. Author
4. Publisher
5. Edition
6. ISBN
7. Genre(FK) – create a separate table for Book genre and use foreign key
8. Language -- either direct value or create FK from separate table
9. Number of copies
10. Shelf Location – Either direct value or FK from separate table
11. Price
12. Status – create a trigger if when book is borrowed status updates to issued/unavailable or -1 from available copies.

**4.Table Book borrowed by Students**

1. Transaction ID
2. Student UID (FK) – From student table
3. Book UID (FK) – From book table
4. Borrowed Date
5. Expected Return date – Create a trigger to add date 7 days after borrowed date
6. Actual Return date
7. Fine – create mechanism which will add daily fine amount for each day book is late
8. Status – enum borrowed, returned, overdue
9. Remarks
10. Librarian ID(FK) – foreign key form librarian table used to check what transactions happened during specific librarian duty.

**5.Table Book borrowed by Staff**

1. Transaction ID
2. Staff UID (FK) – From staff table
3. Book UID (FK) – From book table
4. Borrowed Date
5. Expected Return date – Create a trigger to add date 15 days after borrowed date
6. Actual Return date
7. Status – enum borrowed, returned, overdue
8. Remarks
9. Librarian ID(FK) – foreign key form librarian table used to check what transactions happened during specific librarian duty.

**Q -> Should we create a single table for borrow and return and toggle the status or create separate tables.**

**6. Table for courses**

1. Course ID
2. Course Name
3. Course fees
4. Course Branch
5. Course duration

**7. Table Book Genre**

1. Genre ID
2. Genre Name
3. Description

**8.Table Librarian**

1. LibrarianID
2. Librarian Name
3. Role ID (FK) – Foreign key from table role
4. Gender
5. Age
6. Email
7. Phone no
8. Address

**9.Table Roles**

* RoleID

1. Admin

|  |
| --- |
| 1. Junior Librarian |

|  |
| --- |
|  |

1. Librarian

* Role name
* Role permissions
* Role created date

**10. Table Librarian Credentials**

1. Credential ID
2. LibrarianID (FK) – foreign key from librarian table
3. Username
4. Password

11.Staff branch computer science, commerence, sicence, eng, etc

12.staff faculty professor, hod, co-ordinator

**We can also create system log records table for library**

**Java Backend**

The best architecture to build backend is Layered Architecture with combination of DAO layer.

1. **Presentation Layer (Controller Layer)**

* It receives user requests (like clicking a button or calling an API).
* It passes the request to the Service Layer.

1. **Service Layer**

* It processes the data or performs the required operations.
* It asks the DAO layer to interact with the database.

1. **DAO (Data Access Object Layer)**

* It contains methods to fetch, save, update, or delete data.

1. **Model Layer (Entity Layer)**

* It maps to a database table using JPA.

**How It Works Together**

1. The **Controller** handles the user’s request (like "Get user by ID").
2. It forwards the request to the **Service Layer**.
3. The **Service Layer** asks the **DAO Layer** to fetch or manipulate the database.
4. The **DAO Layer** interacts with the database and returns the data.
5. The **Service Layer** processes the data and sends it back to the **Controller**.
6. The **Controller** returns the response to the user.

**Q -> Should we use Spring data JPA or hibernate for database operations…?**