**Data Sources**

**Primary Data Source: Open Library API**

We will use the Open Library Book Search API and related endpoints to gather comprehensive book data.

**API Endpoints:**

1. **Book Search API**: https://openlibrary.org/search.json?q={query}
2. **Works API**: https://openlibrary.org/works/{work\_id}.json
3. **Authors API**: https://openlibrary.org/authors/{author\_id}.json
4. **ISBN API**: https://openlibrary.org/isbn/{isbn}.json

**Secondary Data Source: Faker Gem**

We'll use the Faker gem to generate additional realistic data for:

* User reviews/ratings
* User profiles
* Reading lists

**Third Data Source: Static CSV**

We'll create a CSV file with popular book categories and reading difficulty levels.

**Data Structure Overview**

**Books (from Open Library API)**

* Title
* ISBN (10 and 13)
* Publication date
* Number of pages
* Description/summary
* Cover image URLs
* Language
* Publisher information

**Authors (from Open Library API)**

* Name
* Bio
* Birth/death dates
* Photos
* Associated works

**Publishers (extracted from book data)**

* Name
* Founded year (if available)

**Categories/Subjects (from CSV + API data)**

* Subject name
* Description
* Difficulty level

**Users (Faker-generated)**

* Name
* Email
* Join date
* Favorite genres

**Reviews (Faker-generated)**

* Rating (1-5)
* Review text
* Review date

**Database Tables Required**

**Primary Tables:**

1. **books** - Store book information
2. **authors** - Store author information
3. **publishers** - Store publisher information
4. **subjects** - Store book categories/subjects
5. **users** - Store user information (Faker)
6. **reviews** - Store book reviews (Faker)

**Join Tables:**

1. **book\_authors** - Many-to-many relationship between books and authors
2. **book\_subjects** - Many-to-many relationship between books and subjects

**Data Relationships**

**One-to-Many Relationships:**

* Publisher → Books (one publisher can have many books)
* User → Reviews (one user can write many reviews)

**Many-to-Many Relationships:**

* Books ↔ Authors (books can have multiple authors, authors can write multiple books)
* Books ↔ Subjects (books can belong to multiple subjects, subjects can contain multiple books)
* Books ↔ Reviews (books can have multiple reviews, handled through book\_id foreign key)

**Data Integration Strategy**

1. **API Data**: Pull book data from Open Library, extract author and publisher information
2. **Normalized Storage**: Store authors and publishers in separate tables to avoid duplication
3. **Join Tables**: Use many-to-many relationships for complex associations
4. **Faker Data**: Generate users and reviews that reference the imported books
5. **CSV Enhancement**: Add category metadata and difficulty levels to enrich the subject data

**Minimum Data Requirements**

* Target: 200+ total rows across all tables
* Estimated breakdown:
  + 50 books from API
  + 30 unique authors
  + 20 publishers
  + 15 subjects/categories
  + 25 users (Faker)
  + 60+ reviews (Faker)

This approach ensures rich, interconnected data suitable for demonstrating Rails associations and complex queries.

