Problem Statement

The gol is to design and develop a **Book Management System** That tracks and manages information about books, authors, editors, suppliers, and items reflated to books. The system should allow a shop owner to manage their inventory, sales, and supply chain. The application needs to support the following families:

- Tracking book details, including it title and subjects.
- Maintaining records of the author who wrote books and editors who edit theem.
- Managing a list of suppliers who provide items (e.g., paper, ink) required for books.
- Recording sales of books by a shop owner.

The system will be sold as the backend for a future application, which will handle book-related operations affixed and with data ingressivity.

Conceptual Design: ER Diagram

The first step in designing the database is crating a **conceptual data model** using an **Entity-Relationship (ER) Diagram**. This diagram services as a blueprint for the database structure. The key compartments of the model are:

- **Entities:** These are the main objects or concpts in the system. The ER diagram identifiers six primary entities: **Book**, **Author**, **Editor**, **Supplier**, **Item**, And **Shop_Owner**. Each is represented by registered.
- **Attributes:** These are the subjects that describe each entity. For example, the Book entity has a Title And a unique Book ID. Author has an Author ID And Name.
- **Relationships:** These show how entities are connected. The diagram shows relationship like:
    - **Writes**: Author writes Book.
    - **Edits**: Editor edits Book.
    - **requires**: Book Requires Item.
    - **Suplies**: Supplier Supplies Item.
    - **Sales**: Shop_Owner Cells Book.
- **Cardinality:** This specifies the number of instances of one entity that can be associated with another. All the main relationship in this model are **many-to-many (M:N)**, mening, for instance, that one Author can write many Books, and one Book can have man Authors. This is crucial for correctly covering the model into the tables.
- **Multi-valued Attributes:** Attributes like Subject (For both Book And Author) are multi-valued, mening a single Book can have multiple subjects (e.g., "Computer Science" and "Data Structures").

Logical Design: Relational Schema & Normalization

The final and most critical step is covering the ER diagram into a **normalized relational schema**. This process translates the conceptual model into a set of structured tables to encure **data integrity** And **minimize redundancy**.

Here's the final normalized schema, derived by applying the results of normalization:

- **Entities to Tables**: Each entity becomes its own table.
- **M:N Relationships & Multi-valued Attributes**: These are handled by crating new, separate **junction tables**. This is the schema of this is in the last **Third Normal Form (3NF)**.

Here are the final normalized tables for your project:

1. **Shop_Owner**
   - Owner_ID(PK)
   - Shop_Name
2. **Book**
   - Book_ID(PK)
   - Title
3. **Author**
   - Author_ID(PK)
   - Name
4. **Editor**
   - Editor_ID(PK)
   - Name
5. **Supplier**
   - Supplier_ID(PK)
   - Name
6. **Item**
   - Item_ID(PK)
   - Item_Name
7. **Book_Author** (Junction Table for the Writes relationship)
   - Book_ID(FK, part of Composite PK)

    - Author_ID(FK, part of Composite PK)

8. **Book_Editor** (Junction Table for the Edits relationship)
    - Book_ID(FK, part of Composite PK)
    - Editor_ID(FK, part of Composite PK)
9. **Book_Sales_Owner** (Junction Table for the Sales relationship)
    - Book_ID(FK, part of Composite PK)
    - Owner_ID(FK, part of Composite PK)
    - Sale_Date
    - Price
10. **Book_Subject** (Junction Table for multi-valued Subject Attribute of Book)
    - Book_ID(FK, part of Composite PK)
    - Subject(Part of Composite PK)
11. **Book_Item** (Junction Table for the requires relationship)
    - Book_ID(FK, part of Composite PK)
    - Item_ID(FK, part of Composite PK)
12. **Item_Supplier** (Junction Table for the Suplies relationship)
    - Item_ID(FK, part of Composite PK)
    - Supplier_ID(FK, part of Composite PK)

This final set of tables represents a robust and afficient detabase structure that is already for implementation as the backend of your application.