# **Experiment 3 :**

**Creating a data dictionary for a student library management system**

**Objective**

This experiment focuses on creating a data dictionary for a student library management system. The data dictionary defines the system’s key components, including entities, attributes, data types, storage requirements, constraints, and relationships. Having a well-structured data dictionary helps maintain data accuracy, consistency, and efficiency, ensuring smooth library operations.

**Introduction**

A data dictionary serves as a structured guide that outlines the details of a database, such as attribute names, data types, and constraints. It plays a vital role in keeping information well-organized and preventing errors.

In a student library management system, the database keeps track of books, students, borrowing transactions, and administrative activities. Clearly defining and structuring the data prevents redundancy, maintains accuracy, and improves system performance. This experiment focuses on developing a well-organized data dictionary that supports efficient library management.

**Methodology**

The main components of the library system include:

* Student – Represents individuals borrowing books.
* Book – Represents books available in the library.
* Transaction – Keeps track of book borrow and return records.
* Librarian – Manages book records and transactions.

For each entity, the following steps are taken:

* Assigning suitable data types to store information efficiently.
* Specifying how much storage is required for each attribute.
* Setting up primary and foreign keys to link related tables.
* Applying constraints like "not null" and "unique" to ensure data reliability.

**Establishing Relationships**

The system follows these basic relationships:

* A student can borrow multiple books.
* A book can only be issued to one student at a time.
* Transactions record when books are borrowed and returned.
* Librarians oversee and manage all transactions.

This approach ensures the library system functions smoothly by keeping records well organized and easily accessible.

**DATA DICTIONARY**

| **Field Name** | **Data Type** | **Storage** | **Description** | **Constraints** |
| --- | --- | --- | --- | --- |
| Student\_ID | integer | 4 | Unique identifier for each student | Primary Key, Not Null |
| Student\_Name | Variable character | 100 | Full name of the student | Not Null |
| Email | Variable character | 100 | Student’s email address | Unique, Not Null |
| PhoneNumber | Variable character | 15 | Student’s contact number | Not Null |
| Course | Variable character | 50 | Course enrolled by the student | Not Null |
| Book\_ID | Integer | 4 | Unique identifier for each book | Primary Key, Not Null |
| BookTitle | Variable character | 255 | Title of the book | Not Null |
| Author | Variable character | 100 | Name of the book’s author | Not Null |
| Category | Variable character | 50 | Genre or category of the book | Not Null |
| ISBN | Variable character | 20 | International Standard Book Number | Unique, Not Null |
| Transaction\_ID | Integer | 4 | Unique transaction identifier | Primary Key, Not Null |
| Book\_ID | Integer | 4 | Book that was borrowed | Foreign Key, References Book Table |
| Student\_ID | Integer | 4 | Student who borrowed the book | Foreign Key, References Student Table |
| Issued\_Date | Date | 3 | Date when the book was issued | Not Null |
| Due\_Date | Date | 3 | Date when the book is due for return | Not Null |
| Return\_date | Date | 3 | Date when the book was returned (if returned) | Nullable |
| FineAmt | Decimal | 5 | Fine amount in case of late return | Default = 0.00 |
| Librarian\_ID | Integer | 4 | ID of the librarian handling the transaction | Foreign Key, References Librarian Table |

## **Defined Relationships**

The relationships between the entities in the Student Library Management System are structured as follows:

1. Student → Transaction (One-to-Many)

○ A student can borrow multiple books at different times.

○ The **StudentID** in the **Transaction** table is a **foreign key** referencing the **Student** table.

1. Book → Transaction (One-to-Many)

○ A book can be borrowed multiple times by different students.

○ The BookID in the Transaction table is a foreign key referencing the Book table.

1. Librarian → Transaction (One-to-Many)

○ A librarian oversees multiple transactions.

○ The LibrarianID in the Transaction table is a foreign key referencing the Librarian table.

1. Student → Book (Many-to-Many through Transactions**)**

○ A student can borrow multiple books, and a book can be issued to different students at different times.

○ This many-to-many relationship is implemented using the Transaction table.

## **Result**

The experiment successfully:

● Identified the key entities in the Student Library Management System.

● Defined attributes, data types, storage requirements, and constraints for each entity.

● Established relationships between entities using primary and foreign keys.

● Developed a structured data dictionary to ensure data integrity and efficient management.