**Problem statement: -**

Creation of a library management database using mysql and performing operations on the database using python.

**DATABASE SCHEMA:-**

**Admins**: attributes: - ids(primary key), name, email, password, mobile\_no.

**Author**: attributes: - author\_id(primary key), author\_name.

**Books**: Attributes: books\_id(Primary Key), books\_name, Author\_id, catogery, books\_nos, books\_price.

**Catogery**: attributes: - catogery\_id(primary key), and catogery\_name.

**users**: Attributes: User\_id (Primary Key), Name, Email, Address, passkey, Phone Number, address.

**Issued\_books:** Attributes:serial\_no(primarykey),book\_nos,books\_name,book\_author,user\_id,status,issue\_date.

**Relationships:**

* **Author - Books (1:N or One-to-Many):** One author can write multiple books. This is represented by the line connecting Author and Books, with "1" on the Author side and "N" on the Books side.
* **Books - Category (1:N or One-to-Many):** One category can have multiple books.
* **Users - Issued Book (1:N or One-to-Many):** One user can issue multiple books (over time, represented by multiple entries in the Issued Book table).
* **Issued Book - Books (1:1 or One-to-One):** Each entry in the Issued Book table represents one specific book being issued. Note: While a book *could* be issued multiple times, each *issuance* is a separate record, hence the 1:1 relationship *between the issuance record and a specific book instance*. This avoids data redundancy (like storing the book name multiple times). The books\_name and book\_author in Issued Book might seem redundant, but they are useful for historical records even if the book's information changes in the Books table.
* **Admins:** The Admins table has no direct relationship with other tables in this simplified ER diagram. It's an independent entity for managing administrative users of the system. In a more complex system, we might have relationships like "Admin adds Book" or "Admin manages User," but those are not represented in the current code.

**DATABASE CONNECTION: -**

The following steps are used to connect a python application to database.

1. Import mysql.connector module (python module used to import mysql).
2. Create a connection object with the database.

Syntax for connection object is: -

Connection\_object() = mysql.connector.connect(host=<host name>,user=<user name>,password=<db password>,database=<database name>).

1. Create the cursor object.
2. Execute the queries.

**Connection: -**

Def connect\_to\_db(): establishes a connection to the mysql database using the ‘mysql.connector’ library. It uses the following connection parameters:

* ‘host’ : ‘host name’ #example= ‘localhost’
* ‘user’ : ’user name’ #example= ‘root’
* ‘password’ : ‘password’ #example = database password(or server password).
* ‘database(db)’ : ‘database name’ #example= ‘library\_management’.

**Data Manipulation Functions –**

**`fetch\_data(query)`:** Executes a given SQL query and fetches all the resulting rows. It handles database connection and cursor management.

**- `insert\_data(query, data)`:** Executes an SQL INSERT query with provided data. It commits the changes to the database.

**- `delete\_data(query, data)`:** Executes an SQL DELETE query with provided data. It commits the changes to the database.

**Display Function –**

**`display\_results(results)`:** Takes a list of rows (results from a query) and prints them in a user-friendly format.

**Menu Functions –**

**`main\_menu()`:** The main program loop. It presents a menu with options to insert, delete, retrieve data, or exit

**. - `insert\_menu()`:** Presents a sub-menu for choosing which table to insert data into (Author, Admins, Books, Category, Users, Issued Books). It prompts the user for the necessary data and constructs the appropriate SQL INSERT query.

**– `delete\_menu()`:** Presents a sub-menu for choosing which table to delete data from. It prompts the user for the ID of the record to delete and constructs the SQL DELETE query.

**- `retrieve\_menu()`:** Presents a sub-menu for choosing which table to retrieve data from. It constructs the SQL SELECT query and displays the results.

**How to Run:**

1. Install MySQL Connector: `pip install mysql-connector-python`

2. Set up MySQL Database: Create a database named `library\_manage` and create the necessary tables with appropriate columns and data types.

3. Update Database Credentials: Modify the `connect\_to\_db()` function with your MySQL username, password, and database name if different.

4. Run the Python script: `python your\_script\_name.py`