Project Name: Library Management System with Database Using Python

Objective

The objective of this project is to create a comprehensive library management system that facilitates the management of students and books within a library. The system allows for:

- Creation and management of student and book records.
- Issuance and return of books.
- Displaying and modifying existing records.
- Ensuring that the database reflects the current status of books and student records.

Summary

The project uses Python and MySQL to implement a library management system with the following features:

- **Database Schema**: The system interacts with a MySQL database consisting of three main tables: students, books, and issued_books.
    - students: Stores student records including roll number, name, and branch.
    - books: Stores book records including book ID, title, author, and availability status.
    - issued_books: Tracks the issuance of books to students with issue and return dates.

Key Functions

1. **Initialize Database**:
    - Creates tables if they don't exist and sets up the database schema.
2. **Student Management**:
    - **Create Student**: Adds a new student record.
    - **Display All Students**: Lists all students and their details.
    - **Display Specific Student**: Shows details for a specific student based on roll number.
    - **Modify Student**: Updates student details.
    - **Delete Student**: Removes a student record.
3. **Book Management**:
    - **Create Book**: Adds a new book record.
    - **Display All Books**: Lists all books and their availability status.
    - **Display Specific Book**: Shows details for a specific book based on book ID.
    - **Modify Book**: Updates book details.
    - **Delete Book**: Removes a book record.

4. **Book Issuance and Return**:
    - **Issue Book**: Records the issuance of a book to a student and updates book availability.
    - **Deposit Book**: Records the return of a book by a student and updates book availability.
5. **Menu System**:
    - Provides a user interface for interacting with the system, including administration options and main operations.

Results

- **Database Schema**: The database is successfully initialized with tables for students, books, and issued books. The schema ensures referential integrity with foreign key constraints.
- **Student and Book Management**: The system effectively manages student and book records with functionality for creating, modifying, displaying, and deleting records.
- **Book Issuance and Return**: The system tracks the issuance and return of books, updating their availability status accordingly.
- **User Interface**: The menu-driven interface allows users to perform various operations easily and navigate between different functionalities.

Conclusion

The Library Management System is a robust solution for managing library operations, including student and book records, and tracking book issuance and returns. The use of Python and MySQL ensures a reliable and efficient system with a user-friendly interface. The system provides comprehensive features to manage library resources effectively, making it a valuable tool for educational institutions or libraries.

Code :

```
import mysql.connector
from datetime import date

# Define the database connection parameters
DB_CONFIG = {
    'user': 'root',
    'password': 'Aditya@8624',
    'host': 'localhost',
    'database': 'library'
}

# Function to initialize the database
def initialize_db():
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute('''CREATE TABLE IF NOT EXISTS students (
                        roll_no VARCHAR(255) PRIMARY KEY,
                        name VARCHAR(255),
                        branch VARCHAR(255)
                      )''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS books (
                        book_id VARCHAR(255) PRIMARY KEY,
                        title VARCHAR(255),
                        author VARCHAR(255),
                        available INT
                      )''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS issued_books (
```

```
                student_roll_no VARCHAR(255),
                book_id VARCHAR(255),
                issue_date DATE,
                return_date DATE,
                FOREIGN KEY(student_roll_no) REFERENCES students(roll_no),
                FOREIGN KEY(book_id) REFERENCES books(book_id)
            )''')

    conn.commit()
    conn.close()

# Function to create student record
def create_student():
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    print("\nEnter student details:")
    roll_no = input("Roll No: ")
    name = input("Name: ")
    branch = input("Branch: ")

    cursor.execute("INSERT INTO students (roll_no, name, branch) VALUES (%s, %s, %s)", (roll_no,
name, branch))

    conn.commit()
    conn.close()

    print(f"\nStudent Record for Roll No {roll_no} created successfully.")

# Function to display all students
def display_all_students():
```

```
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM students")
    students = cursor.fetchall()

    print("\nAll Students:")
    for student in students:
        print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")

    print(f"Total Students: {len(students)}")

    conn.close()

# Function to display specific student record
def display_specific_student():
    roll_no = input("\nEnter Roll No of the student to display: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM students WHERE roll_no = %s", (roll_no,))
    student = cursor.fetchone()

    if student:
        print("\nStudent Details:")
        print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")
    else:
        print(f"\nStudent with Roll No {roll_no} not found.")

    conn.close()
```

```
# Function to modify student record
def modify_student():
    roll_no = input("\nEnter Roll No of the student to modify: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM students WHERE roll_no = %s", (roll_no,))
    student = cursor.fetchone()

    if student:
        print("\nCurrent Student Details:")
        print(f"Roll No: {student[0]}, Name: {student[1]}, Branch: {student[2]}")
        print("\nEnter new details:")
        name = input("Name (leave blank to keep current): ").strip()
        branch = input("Branch (leave blank to keep current): ").strip()

        if name:
            cursor.execute("UPDATE students SET name = %s WHERE roll_no = %s", (name, roll_no))
        if branch:
            cursor.execute("UPDATE students SET branch = %s WHERE roll_no = %s", (branch, roll_no))

        conn.commit()
        print(f"\nStudent Record for Roll No {roll_no} modified successfully.")
    else:
        print(f"\nStudent with Roll No {roll_no} not found.")

    conn.close()

# Function to delete student record
```

```
def delete_student():
    roll_no = input("\nEnter Roll No of the student to delete: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("DELETE FROM students WHERE roll_no = %s", (roll_no,))
    conn.commit()

    if cursor.rowcount > 0:
        print(f"\nStudent Record for Roll No {roll_no} deleted successfully.")
    else:
        print(f"\nStudent with Roll No {roll_no} not found.")

    conn.close()

# Function to create book
def create_book():
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    print("\nEnter book details:")
    book_id = input("Book ID: ")
    title = input("Title: ")
    author = input("Author: ")

    cursor.execute("INSERT INTO books (book_id, title, author, available) VALUES (%s, %s, %s, 1)",
(book_id, title, author))

    conn.commit()
    conn.close()
```

```
    print(f"\nBook Record for Book ID {book_id} created successfully.")

# Function to display all books
def display_all_books():
    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM books")
    books = cursor.fetchall()

    print("\nAll Books:")
    for book in books:
        status = "Available" if book[3] == 1 else "Issued"
        print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}, Status: {status}")

    print(f"Total Books: {len(books)}")

    conn.close()

# Function to display specific book record
def display_specific_book():
    book_id = input("\nEnter Book ID of the book to display: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM books WHERE book_id = %s", (book_id,))
    book = cursor.fetchone()

    if book:
```

```
        status = "Available" if book[3] == 1 else "Issued"
        print("\nBook Details:")
        print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}, Status: {status}")
    else:
        print(f"\nBook with Book ID {book_id} not found.")

    conn.close()

# Function to modify book record
def modify_book():
    book_id = input("\nEnter Book ID of the book to modify: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM books WHERE book_id = %s", (book_id,))
    book = cursor.fetchone()

    if book:
        print("\nCurrent Book Details:")
        print(f"Book ID: {book[0]}, Title: {book[1]}, Author: {book[2]}")
        print("\nEnter new details:")
        title = input("Title (leave blank to keep current): ").strip()
        author = input("Author (leave blank to keep current): ").strip()

        if title:
            cursor.execute("UPDATE books SET title = %s WHERE book_id = %s", (title, book_id))
        if author:
            cursor.execute("UPDATE books SET author = %s WHERE book_id = %s", (author, book_id))

        conn.commit()
```

```
        print(f"\nBook Record for Book ID {book_id} modified successfully.")
    else:
        print(f"\nBook with Book ID {book_id} not found.")

    conn.close()

# Function to delete book record
def delete_book():
    book_id = input("\nEnter Book ID of the book to delete: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("DELETE FROM books WHERE book_id = %s", (book_id,))
    conn.commit()

    if cursor.rowcount > 0:
        print(f"\nBook Record for Book ID {book_id} deleted successfully.")
    else:
        print(f"\nBook with Book ID {book_id} not found.")

    conn.close()

# Function to issue a book
def issue_book():
    student_roll_no = input("\nEnter Student Roll No: ")
    book_id = input("Enter Book ID: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()
```

```
    cursor.execute("SELECT * FROM students WHERE roll_no = %s", (student_roll_no,))
    student_exists = cursor.fetchone() is not None

    cursor.execute("SELECT * FROM books WHERE book_id = %s", (book_id,))
    book = cursor.fetchone()

    book_exists = book is not None
    book_available = book_exists and book[3] == 1

    if not student_exists:
        print(f"\nStudent with Roll No {student_roll_no} not found.")
        conn.close()
        return

    if not book_exists:
        print(f"\nBook with Book ID {book_id} not found.")
        conn.close()
        return

    if not book_available:
        print(f"\nBook with Book ID {book_id} is already issued.")
        conn.close()
        return

    cursor.execute("INSERT INTO issued_books (student_roll_no, book_id, issue_date, return_date)
VALUES (%s, %s, %s, %s)",
                   (student_roll_no, book_id, date.today(), None))
    cursor.execute("UPDATE books SET available = 0 WHERE book_id = %s", (book_id,))

    conn.commit()
    conn.close()
```

```
    print(f"\nBook ID {book_id} issued to Student Roll No {student_roll_no} successfully.")

# Function to deposit a book
def deposit_book():
    student_roll_no = input("\nEnter Student Roll No: ")
    book_id = input("Enter Book ID: ")

    conn = mysql.connector.connect(**DB_CONFIG)
    cursor = conn.cursor()

    cursor.execute("SELECT * FROM issued_books WHERE student_roll_no = %s AND book_id = %s
AND return_date IS NULL",
                   (student_roll_no, book_id))
    issued_book = cursor.fetchone()

    if not issued_book:
        print(f"\nNo record found for Book ID {book_id} issued to Student Roll No {student_roll_no}.")
        conn.close()
        return

    cursor.execute("UPDATE issued_books SET return_date = %s WHERE student_roll_no = %s AND
book_id = %s AND return_date IS NULL",
                   (date.today(), student_roll_no, book_id))
    cursor.execute("UPDATE books SET available = 1 WHERE book_id = %s", (book_id,))

    conn.commit()
    conn.close()

    print(f"\nBook ID {book_id} returned by Student Roll No {student_roll_no} successfully.")

# Main function to display menu and process user input
```

```
def main():
    initialize_db()

    while True:
        print("\n***** LIBRARY MANAGEMENT SYSTEM *****")
        print("1. BOOK ISSUE")
        print("2. BOOK DEPOSIT")
        print("3. ADMINISTRATION MENU")
        print("4. EXIT")

        choice = input("\nEnter your choice: ")

        if choice == '1':
            issue_book()
        elif choice == '2':
            deposit_book()
        elif choice == '3':
            print("\n***** ADMINISTRATION MENU *****")
            print("a. CREATE STUDENT RECORD")
            print("b. DISPLAY ALL STUDENTS RECORD")
            print("c. DISPLAY SPECIFIC STUDENT RECORD")
            print("d. MODIFY STUDENT RECORD")
            print("e. DELETE STUDENT RECORD")
            print("f. CREATE BOOK")
            print("g. DISPLAY ALL BOOKS")
            print("h. DISPLAY SPECIFIC BOOK")
            print("i. MODIFY BOOK RECORD")
            print("j. DELETE BOOK RECORD")
            print("k. BACK TO MAIN MENU")

            admin_choice = input("\nEnter your choice: ")
```

```
        if admin_choice == 'a':
            create_student()
        elif admin_choice == 'b':
            display_all_students()
        elif admin_choice == 'c':
            display_specific_student()
        elif admin_choice == 'd':
            modify_student()
        elif admin_choice == 'e':
            delete_student()
        elif admin_choice == 'f':
            create_book()
        elif admin_choice == 'g':
            display_all_books()
        elif admin_choice == 'h':
            display_specific_book()
        elif admin_choice == 'i':
            modify_book()
        elif admin_choice == 'j':
            delete_book()
        elif admin_choice == 'k':
            continue
        else:
            print("\nInvalid choice! Please enter a valid option (a-k).")
    elif choice == '4':
        print("\nThank you for using the Library Management System. Goodbye!")
        break
    else:
        print("\nInvalid choice! Please enter a valid option (1-4).")
```

```
# Entry point of the program
if __name__ == "__main__":
    main()
```