Topic: Library Management System

StudentID: 23073192

Name:- Sravani Prathipati

Emailid :- ps24aan@herts.ac.uk

Introduction :

A software program called a Library Management System (LMS) is made to automate and simplify a library's essential processes. In order to provide customers and library employees with an effective and well-organised experience, it controls users, resources, and transactional operations. The LMS optimises resource use and guarantees that library activities are completed precisely and on schedule by combining these three essential elements.

1.Users

- Users are patrons or members of the library. They are the main users who engage with the system in order to utilise library services. Essential user information is stored by the LMS in order to keep correct records and offer individualised assistance.

2.Books

- The main resources of an LMS are books, or library materials. Because of their systematic cataloguing, users can find, borrow, and reserve these resources with ease. This organisation facilitates effective resource access and discovery.

3.Transactions:

- Borrowing and returning materials is one of the LMS's primary functions. To guarantee equity, uphold accountability, and encourage the appropriate sharing of library resources, these exchanges are closely monitored and controlled.

The Interaction of Users, Books, and Transactions

The LMS seamlessly combines users, books, and transactions to produce a seamless and effective workflow.

- Patrons can view and borrow available titles by perusing the library's catalogue. The system records this activity as a transaction.

- The LMS tracks books and, when an item is checked out, immediately updates the user's loan record and modifies the number of available copies.

- The borrowing procedure is managed by administrators; employees can keep an eye on past-due items, compute late return penalties, and provide reports to examine borrowing trends.

About the data generation

I Using Python, I created each table and dataset, saved it as a distinct CSV file in my downloads folder, and then imported it into SQLite. The following Python libraries were used in the data creation process:

- Faker: This library assisted in producing accurate data for text-based fields such as names and dates.
- NumPy: A tool for generating numerical data and random values for a range of categories, including prices, earnings, and age groupings.
- Pandas: Used to examine the data and make any necessary extra changes or improvements.

Data for Each Table :

1. User table

Data pertaining to library users is intended to be stored and arranged in the Users table. It contains a number of fields that aid in maintaining each person's critical information and helping them be individually identified.

The UserID column acts as the primary key, giving each user a unique identification. This guarantees that every entry is distinct and permits appropriate referencing between other database tables. The Name field contains the user's entire name, and the Email column has their email address. Both of these fields are designated as NOT NULL, meaning that they must be present in every entry because they are essential for identification and communication.

In addition to providing helpful demographic information, the Age column helps enforce age-related limitations on specific content or services by recording the user's age as an integer.

Classification of Data Types:

Nominal Information: Name

Ordinal Information: Type of Membership

Data Interval: Date of Registration

The OutstandingFees ratio data

```sql
-- Creating the Users table
CREATE TABLE Users (
    UserID INTEGER PRIMARY KEY,     -- Unique identifier for each user
    Name TEXT NOT NULL,             -- User's name
    Email TEXT NOT NULL,            -- User's email
    Age INTEGER,                    -- User's age
    Gender TEXT,                    -- User's gender
    JoinDate DATE,                  -- The date the user joined
    MembershipLevel TEXT            -- Membership level of the user
);
```

	UserID	Name	Email	Age	Gender	JoinDate	MembershipLevel
1	1	Jimmy Hurst	fmoyer@example.net	33	Other	2000-12-05	Basic
2	2	Ronald Lee DVM	arnoldlisa@example.org	66	Female	2006-11-04	Gold
3	3	Jillian Levy	jonesjulie@example.org	68	Non-Binary	2016-08-02	Premium
4	4	Catherine Villa	ruizdaniel@example.com	46	Male	2009-06-06	Premium
5	5	Timothy Rodriguez	webbtheresa@example.com	75	Female	2023-08-14	Basic
6	6	Jessica King	eanderson@example.net	75	Male	2017-11-28	Basic
7	7	Robert Marshall	marcwarner@example.com	19	Female	2017-05-26	Basic

```
Execution finished without errors.
Result: 500 rows returned in 5ms
At line 35:
Select * from users
```

2. Books table

The Books table is made to hold comprehensive data about the library's collection of books, facilitating effective resource management and tracking. The primary key is the BookID column, which gives each item a unique identity and guarantees that each book in the collection has a unique identification. This is essential for precisely tracking and overseeing certain objects in the library's collection.

The book's title and the author's name are recorded in the Title and Author columns, respectively. Since both fields are set to NOT NULL, each book record must have this information. These qualities are essential for finding books, keeping the catalogue organised, and facilitating efficient search capabilities.

To help with content filtering and user preferences, the Genre column divides each book into genres like fiction, non-fiction, or science fiction.

Classification of Data Types:

Nominal Information: Author, Title, and Genre

Data Interval: PublishDate

Ratio Information: Cost, Copies Available

```sql
-- Creating the Books table
CREATE TABLE Books (
    BookID INTEGER PRIMARY KEY,     -- Unique identifier for each book
    Title TEXT NOT NULL,            -- Title of the book
    Author TEXT NOT NULL,           -- Author of the book
    Genre TEXT,                     -- Genre of the book
    Price REAL,                     -- Price of the book
    PublishDate DATE,               -- Date the book was published
    CopiesAvailable INTEGER         -- Number of copies available
);
```

	BookID	Title	Author	Genre	Price	PublishDate	CopiesAvailable
1	1	The Amazing Journey	David Miller	Non-Fiction	39.27	1955-11-06	7
2	2	The Fascinating Saga	Sarah Miller	Science Fiction	21.06	1935-04-17	8
3	3	The Boring Mystery	David Jones	Science Fiction	25.67	1904-11-15	19
4	4	The Boring Mystery	Emily Williams	Mystery	11.24	1948-02-11	3
5	5	The Boring Story	Sarah Jones	Science Fiction	10.49	2004-10-30	6
6	6	The Wonderful Mystery	Sarah Miller	Biography	13.24	1928-02-18	7
7	7	The Wonderful Journey	Alice Miller	History	44.05	1970-11-11	12

```
Execution finished without errors.
Result: 600 rows returned in 13ms
At line 35:
Select * from books
```

3. Borrow transctions

An essential part of the library management system is the Transactions table, which records all user and book borrowing and return events. The TransactionID column acts as the primary key, uniquely identifying each transaction and guaranteeing that each record is unique and traceable for accountability and future reference.

The UserID field identifies the person involved in the transaction by connecting each transaction to a particular user in the Users table. The BookID column, which details the precise book being borrowed or returned, also links to the Books table. Foreign key constraints, which preserve data integrity by guaranteeing that only legitimate users and books cited in their respective tables are participating in the transaction, are used to maintain these linkages.

The BorrowDate column records the exact date the book was issued, while additional fields track other relevant timing and fee information.

Data Type Classification:

Nominal Data: TransactionID

Interval Data: BorrowDate, ReturnDate

Ratio Data: LateFee

```
22
23    -- Creating the Transactions table
24    CREATE TABLE Transactions (
25        TransactionID INTEGER PRIMARY KEY, -- Unique transaction ID
26        UserID INTEGER,                    -- User involved in the transaction
27        BookID INTEGER,                    -- Book involved in the transaction
28        BorrowDate DATE,                   -- Date the book was borrowed
29        ReturnDate DATE,                   -- Date the book was returned
30        LateFee REAL,                      -- Late fee for the transaction
31        FOREIGN KEY (UserID) REFERENCES Users(UserID),
```

	TransactionID	UserID	BookID	BorrowDate	ReturnDate	LateFee
1	1	444	217	2024-09-03	*NULL*	*NULL*
2	2	219	142	2024-07-02	2024-07-30	*NULL*
3	3	379	395	2024-02-25	2024-03-03	*NULL*
4	4	285	281	2024-08-18	2024-09-14	1.15
5	5	308	385	2024-03-13	2024-04-07	3.58
6	6	204	368	2024-09-11	2024-09-19	*NULL*
7	7	249	174	2024-05-26	2024-06-18	0.2

```
Execution finished without errors.
Result: 1000 rows returned in 127ms
At line 26:
select * from transactions
```

Schema of Database

1. Users Table:

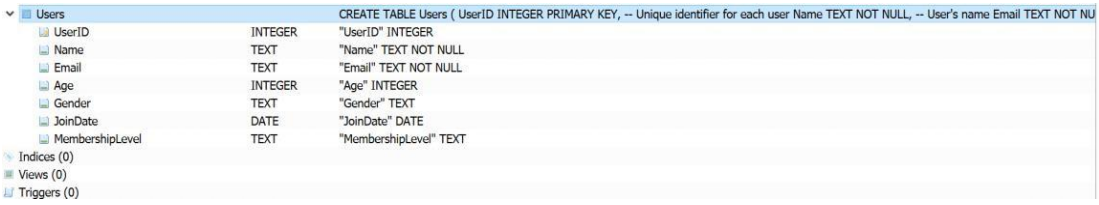
Stores information about library members.

- Primary Key: UserID
- Constraints

Email is unique to prevent duplicates.

Gender is limited to specific values.

- Columns: UserID, Name, Email, Age, Gender, JoinDate, MembershipLevel.



2. Books Table:

Stores details of books in the library.

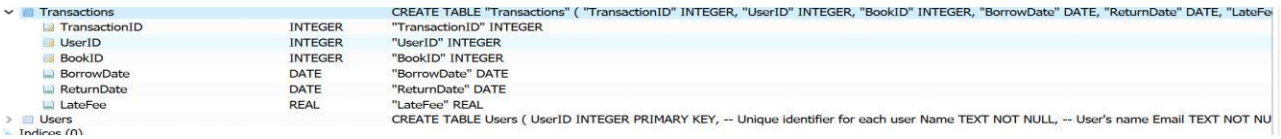
- Primary Key: BookID
- Columns: BookID, Title, Author, Genre, Price, PublishDate, CopiesAvailable.

Name	Type	Schema
Tables (3)		
Books		CREATE TABLE "Books" ("BookID" INTEGER, "Title" TEXT NOT NULL, "Author" TEXT NOT NULL, "Genre" TEXT, "Price" REAL, "PublishDate" DATE, "C
BookID	INTEGER	"BookID" INTEGER
Title	TEXT	"Title" TEXT NOT NULL
Author	TEXT	"Author" TEXT NOT NULL
Genre	TEXT	"Genre" TEXT
Price	REAL	"Price" REAL
PublishDate	DATE	"PublishDate" DATE
CopiesAvailable	INTEGER	"CopiesAvailable" INTEGER

3. Transactions Table:

Tracks borrowing transactions.

- Primary Key: TransactionID
- Foreign Keys: UserID, BookID
- Columns: TransactionID, UserID, BookID, BorrowDate, ReturnDate, LateFee

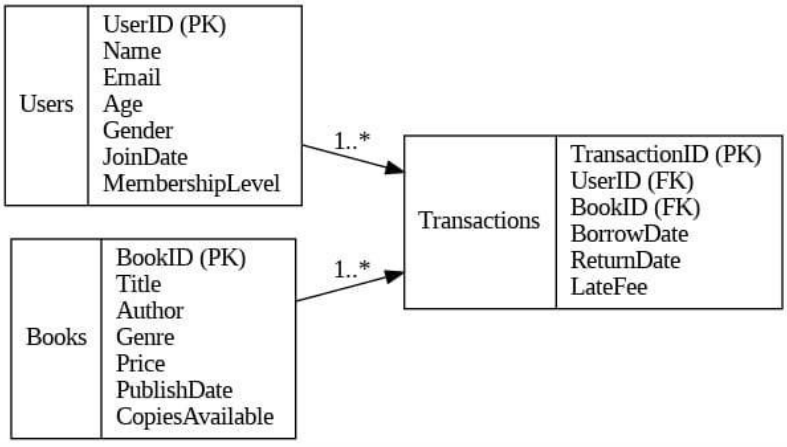


Transactions		CREATE TABLE "Transactions" ("TransactionID" INTEGER, "UserID" INTEGER, "BookID" INTEGER, "BorrowDate" DATE, "ReturnDate" DATE, "LateFe
TransactionID	INTEGER	"TransactionID" INTEGER
UserID	INTEGER	"UserID" INTEGER
BookID	INTEGER	"BookID" INTEGER
BorrowDate	DATE	"BorrowDate" DATE
ReturnDate	DATE	"ReturnDate" DATE
LateFee	REAL	"LateFee" REAL
Users		CREATE TABLE Users (UserID INTEGER PRIMARY KEY, -- Unique identifier for each user Name TEXT NOT NULL, -- User's name Email TEXT NOT NU
Indices (0)		

I have added the Entity-Relationship Diagram to visually

represent the schema and the screenshot of the SQLite. This will give a clear view

of the database.



Justification for database

The Library Management System (LMS) database was organised with distinct tables for Users, Books, and Transactions in order to guarantee data normalisation, improve system efficiency, and preserve data accuracy. The following provides a thorough justification for this table separation:

All pertinent data on library patrons is stored in the Users table, which was created especially for that purpose. It records important information such the user's complete name, email address, membership type, and demographics like gender and age. Clear organisation and simpler maintenance are encouraged when user-related data is separated into its own table.

A thorough catalogue of all the resources in the library is kept up to date by the Books table. Accurate cataloguing and streamlined inventory management are supported by keeping book-specific information in a separate table.

Between the Users and Books tables, the Transactions table serves as a bridge. It creates a comprehensive borrowing history by recording all lending activities, including which user borrowed which book and when.

By splitting data into these logically separate tables, consistency is ensured and redundancy is reduced. For example, the Transactions table just uses keys to refer to the books and user information, eliminating the need to replicate them for each loan.

Effective updates are also made possible by this framework. Without affecting already-existing transaction records, changes to user contact information or book availability can be made directly in the corresponding tables.

Last but not least, this kind of data organisation makes it possible to run quicker and more accurate searches, such seeing a user's borrowing history or determining the status of a particular book.

Note:-

Googlecolablink:-
https://colab.research.google.com/drive/1S_c6jZzFCFNa7u_4OKdAGV3kgeIzP46U?usp=sharing

DBbrowserSQLitefile:-https://github.com/Sravani8787/SQL_Assignment