CS504 FINAL PROJECT

Krishna Sai Pendem

PUBLIC LIBRARY MANAGEMENT SYSTEM

This project's objective is to create a highly efficient database for a library system, enabling it to effectively handle its resources, members, and borrowing operations. The system's main function is to maintain comprehensive records of library materials, including books, magazines, e-books, and audiobooks. Additionally, it will facilitate members in borrowing and reserving materials, while empowering the library staff to oversee resources and assist members efficiently.

The entities and their attributes are as follows:

- Material entity represents individual items available in the library, such as books, magazines, e-books, and audiobooks. A material belongs to a genre and has a catalog entry.

  Attributes of this Entity are:
  - **Material_ID**: A unique identifier for each material.
  - **Title**: The title of the material.
  - **Publication_Date**: The date of publication of the material.
  - **Catalog_ID**: A reference to the catalog entry for the material.
  - **Genre_ID**: A reference to the genre of the material.
- Catalog entity represents a record of library materials with information on their availability and location. A catalog entry has a unique identifier, a name, and a location within the library.

  Attributes of this Entity are:
  - **Catalog_ID**: A unique identifier for each catalog entry.
  - **Name**: The name of the catalog.
  - **Location**: The location of the material within the library.
- Genre entity represents the various genres or categories of library materials. A genre has a unique identifier, a name, and a description.

  Attributes of this Entity are:
  - **Genre_ID**: A unique identifier for each genre.
  - **Name**: The name of the genre.
  - **Description**: The brief introduction of the genre
- Borrow entity represents the borrowing activity of library materials by members. A borrowed transaction has a unique identifier, a reference to the borrowed material, a reference to the member who borrowed the material, a reference to the staff who processed the transaction, and dates for when the material was borrowed, due, and returned.

  Attributes of this Entity are:
  - **Borrow_ID**: A unique identifier for each borrowing transaction.
  - **Material_ID**: A reference to the borrowed material.
  - **Member_ID**: A reference to the member who borrowed the material.
  - **Staff_ID**: A reference to the staff who processed the transaction.

- **Borrow_Date**: The date the material was borrowed.
- **Due_Date**: The date the material is due.
- **Return_Date**: The date the material is returned.

• Author entity represents authors who have created library materials. An author has a unique identifier, a name, a birth date, and a nationality.

Attributes of this Entity are:

- **Author_ID**: A unique identifier for each author.
- **Name**: The name of the author.
- **Birth_Date**: The birth date of the author.
- **Nationality**: The nationality of the author.

• Authorship entity represents the relationship between authors and the materials they have created. An authorship record has a unique identifier, a reference to the author, and a reference to the material authored.

Attributes of this Entity are:

- **Authorship_ID**: A unique identifier for each authorship record.
- **Author_ID**: A reference to the author.
- **Material_ID**: A reference to the material authored.

• Member entities represent library members who can borrow and reserve materials. A member has a unique identifier, a name, contact information, and a join date.

Attributes of this Entity are:

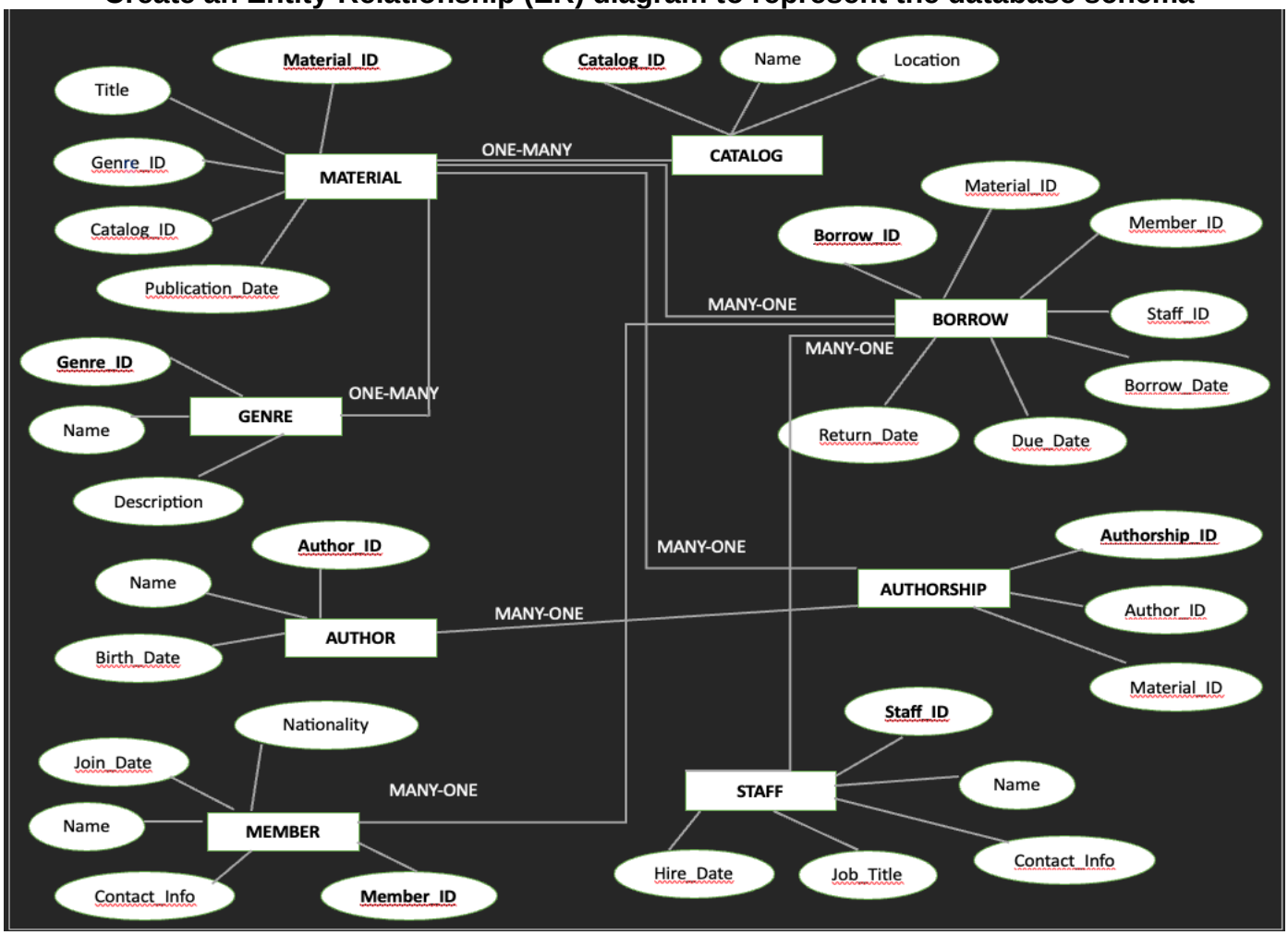
- **Member_ID**: A unique identifier for each member.
- **Name**: The name of the member.
- **Contact_Info**: Email address (or phone number) of the member.
- **Join_Date**: The date the member joined the library

• Staff entity represents library staff who manage library resources and assist members. A staff member has a unique identifier, a name, contact information, a job title, and a hire date.

Attributes of this Entity are:

- **Staff_ID**: A unique identifier for each staff member.
- **Name**: The name of the staff member.
- **Contact_Info**: Email address (or phone number) of the member.
- **Job_Title**: The job title of the staff member (e.g., librarian, assistant librarian).
- **Hire_Date**: The date the staff member was hired by the library.

Create an Entity-Relationship (ER) diagram to represent the database schema



The ER diagram

MATERIAL: This entity represents any type of material that can be borrowed from the library. Each material has a unique Material ID.

CATALOG: This entity contains information about each material, such as the title, genre, and publication date. Each material has a unique Catalog ID.

GENRE: This entity represents the different genres of materials that the library carries. Each genre has a unique Genre ID.

AUTHOR: This entity represents the authors of the materials in the library. Each author has a unique Author ID.

AUTHORSHIP: This entity represents the relationship between an author and a material. Each authorship has a unique Authorship ID.

MEMBER: This entity represents the members of the library. Each member has a unique Member ID.

STAFF: This entity represents the staff members of the library. Each staff member has a unique Staff ID.

BORROW: This entity represents the borrowing of materials by members. Each borrow has a unique Borrow ID.

The relationships between the entities are as follows:

One-to-many relationship between MATERIAL and CATALOG: Each material can have only one catalog entry, but a catalog entry can be associated with multiple materials. This is because a material can have multiple editions or different formats (e.g., a book can be available in hardcover, paperback, and audiobook formats).

One-to-many relationship between GENRE and MATERIAL: Each material can have only one genre, but a genre can be associated with multiple materials.

Many-to-many relationship between AUTHOR and MATERIAL: A material can have multiple authors, and an author can write multiple materials. This relationship is represented by the AUTHORSHIP entity.

Many-to-one relationship between MEMBER and BORROW: A member can borrow multiple materials, but a borrow can only be associated with one member.

Many-to-one relationship between STAFF and BORROW: A staff member can check out materials on behalf of members, but a borrow can only be associated with one staff member.

The ER diagram also shows the following attributes for each entity:

MATERIAL: Material ID, Catalog ID, Name, Location, Title, Genre ID
CATALOG: Catalog ID, Publication Date
GENRE: Genre ID, Name
AUTHOR: Author ID, Name, Birth Date
AUTHORSHIP: Authorship ID, Author ID, Material ID
MEMBER: Member ID, Name, Contact Info
STAFF: Staff ID, Name, Contact Info, Hire Date
BORROW: Borrow ID, Member ID, Material ID, Staff ID, Borrow Date, Return Date, Due Date

This ER diagram can be used to design a database for a library management system. The database would have tables for each of the entities, and the relationships between the entities would be implemented using foreign keys.

1. Choose an appropriate Database Management System (DBMS) for the project.

I have used PostgreSQL and PGadmin for the project.

2. Designing Database and adding Tables.

I have created Database using PGadmin GUI and named as “CS504_Final_Project_Public_library”.

Created Table Staff

```
-- Table for Staff entity with its attributes
CREATE TABLE Staff (
    Staff_ID INTEGER PRIMARY KEY,
    Staff_Name TEXT NOT NULL,
    Contact_Info TEXT NOT NULL,
    Job_Title TEXT NOT NULL,
    Hire_Date DATE NOT NULL
);
```

Inserted Data in Staff Table

```
-- Insert into Staff
INSERT INTO Staff (Staff_ID, Staff_Name, Contact_Info, Job_Title, Hire_Date)
VALUES
    (1, 'Amy Green', 'amy.green@email.com', 'Librarian', '2017-06-01'),
    (2, 'Brian Taylor', 'brian.taylor@email.com', 'Library Assistant', '2018-11-15'),
    (3, 'Christine King', 'chris.king@email.com', 'Library Assistant', '2019-05-20'),
    (4, 'Daniel Wright', 'dan.wright@email.com', 'Library Technician', '2020-02-01');

SELECT * FROM Staff;
```

Output

	staff_id [PK] integer	staff_name text	contact_info text	job_title text	hire_date date
1	1	Amy Green	amy.green@email.com	Librarian	2017-06-01
2	2	Brian Taylor	brian.taylor@email.com	Library Assistant	2018-11-15
3	3	Christine King	chris.king@email.com	Library Assistant	2019-05-20
4	4	Daniel Wright	dan.wright@email.com	Library Technician	2020-02-01

Created Table Author

```
-- Table for Author entity with its attributes
CREATE TABLE Author (
    Author_ID INTEGER PRIMARY KEY,
    Author_Name TEXT NOT NULL,
    Birth_Date DATE,
    Nationality TEXT
);
```

Inserted Data in Author Table

```sql
-- Insert into Author
INSERT INTO Author (Author_ID, Author_Name, Birth_Date, Nationality) VALUES
(1, 'Jane Austen', '1775-12-16', 'British'),
(2, 'Ernest Hemingway', '1899-07-21', 'American'),
(3, 'George Orwell', '1903-06-25', 'British'),
(4, 'Scott Fitzgerald', '1896-09-24', 'American'),
(5, 'J.K. Rowling', '1965-07-31', 'British'),
(6, 'Mark Twain', '1835-11-30', 'American'),
(7, 'Leo Tolstoy', '1828-09-09', 'Russian'),
(8, 'Virginia Woolf', '1882-01-25', 'British'),
(9, 'Gabriel Márquez', '1927-03-06', 'Colombian'),
(10, 'Charles Dickens', '1812-02-07', 'British'),
(11, 'Harper Lee', '1926-04-28', 'American'),
(12, 'Oscar Wilde', '1854-10-16', 'Irish'),
(13, 'William Shakespeare', '1564-04-26', 'British'),
(14, 'Franz Kafka', '1883-07-03', 'Czech'),
(15, 'James Joyce', '1882-02-02', 'Irish'),
(16, 'J.R.R. Tolkien', '1892-01-03', 'British'),
(17, 'Emily Bronte', '1818-07-30', 'British'),
(18, 'Toni Morrison', '1931-02-18', 'American'),
(19, 'Fyodor Dostoevsky', '1821-11-11', 'Russian'),
(20, 'Lucas Piki', '1847-10-16', 'British');

SELECT * FROM Author;
```

Output

	author_id [PK] integer	author_name text	birth_date date	nationality text
1	1	Jane Austen	1775-12-16	British
2	2	Ernest Hemingway	1899-07-21	American
3	3	George Orwell	1903-06-25	British
4	4	Scott Fitzgerald	1896-09-24	American
5	5	J.K. Rowling	1965-07-31	British
6	6	Mark Twain	1835-11-30	American
7	7	Leo Tolstoy	1828-09-09	Russian
8	8	Virginia Woolf	1882-01-25	British
9	9	Gabriel Márquez	1927-03-06	Colombian
10	10	Charles Dickens	1812-02-07	British
11	11	Harper Lee	1926-04-28	American
12	12	Oscar Wilde	1854-10-16	Irish
13	13	William Shakespeare	1564-04-26	British
14	14	Franz Kafka	1883-07-03	Czech
15	15	James Joyce	1882-02-02	Irish
16	16	J.R.R. Tolkien	1892-01-03	British
17	17	Emily Bronte	1818-07-30	British
18	18	Toni Morrison	1931-02-18	American
19	19	Fyodor Dostoevsky	1821-11-11	Russian
20	20	Lucas Piki	1847-10-16	British
21	21	Lucas Pipi	[null]	[null]

Created Table Member

```sql
-- Table for Member entity with its attributes
CREATE TABLE Member (
    Member_ID INTEGER PRIMARY KEY,
    Member_Name TEXT NOT NULL,
    Contact_Info TEXT NOT NULL,
    Join_Date DATE NOT NULL
);
```

Inserted Member Table Data

```sql
--Insert Into Member
INSERT INTO Member (Member_ID, Member_Name, Contact_Info, Join_Date)
VALUES
(1, 'Alice Johnson', 'alice.johnson@email.com', '2018-01-10'),
(2, 'Bob Smith', 'bob.smith@email.com', '2018-03-15'),
(3, 'Carol Brown', 'carol.brown@email.com', '2018-06-20'),
(4, 'David Williams', 'david.williams@email.com', '2018-09-18'),
(5, 'Emily Miller', 'emily.miller@email.com', '2019-02-12'),
(6, 'Frank Davis', 'frank.davis@email.com', '2019-05-25'),
(7, 'Grace Wilson', 'grace.wilson@email.com', '2019-08-15'),
(8, 'Harry Garcia', 'harry.garcia@email.com', '2019-11-27'),
(9, 'Isla Thomas', 'isla.thomas@email.com', '2020-03-04'),
(10, 'Jack Martinez', 'jack.martinez@email.com', '2020-07-01'),
(11, 'Kate Anderson', 'kate.anderson@email.com', '2020-09-30'),
(12, 'Luke Jackson', 'luke.jackson@email.com', '2021-01-18'),
(13, 'Mia White', 'mia.white@email.com', '2021-04-27'),
(14, 'Noah Harris', 'noah.harris@email.com', '2021-07-13'),
(15, 'Olivia Clark', 'olivia.clark@email.com', '2021-10-05'),
(16, 'Peter Lewis', 'peter.lewis@email.com', '2021-12-01'),
(17, 'Quinn Hall', 'quinn.hall@email.com', '2022-02-28'),
(18, 'Rachel Young', 'rachel.young@email.com', '2022-06-17'),
(19, 'Sam Walker', 'sam.walker@email.com', '2022-09-25'),
(20, 'Tiffany Allen', 'tiffany.allen@email.com', '2022-12-10');

SELECT * FROM Member;
```

Output:

	member_id [PK] integer	member_name text	contact_info text	join_date date
1	1	Alice Johnson	alice.johnson@email.com	2018-01-10
2	2	Bob Smith	bob.smith@email.com	2018-03-15
3	3	Carol Brown	carol.brown@email.com	2018-06-20
4	4	David Williams	david.williams@email.com	2018-09-18
5	6	Frank Davis	frank.davis@email.com	2019-05-25
6	7	Grace Wilson	grace.wilson@email.com	2019-08-15
7	8	Harry Garcia	harry.garcia@email.com	2019-11-27
8	9	Isla Thomas	isla.thomas@email.com	2020-03-04
9	10	Jack Martinez	jack.martinez@email.com	2020-07-01
10	11	Kate Anderson	kate.anderson@email.com	2020-09-30
11	12	Luke Jackson	luke.jackson@email.com	2021-01-18
12	13	Mia White	mia.white@email.com	2021-04-27
13	14	Noah Harris	noah.harris@email.com	2021-07-13
14	15	Olivia Clark	olivia.clark@email.com	2021-10-05
15	16	Peter Lewis	peter.lewis@email.com	2021-12-01
16	17	Quinn Hall	quinn.hall@email.com	2022-02-28
17	18	Rachel Young	rachel.young@email.com	2022-06-17
18	19	Sam Walker	sam.walker@email.com	2022-09-25
19	20	Tiffany Allen	tiffany.allen@email.com	2022-12-10

Created Genre Table

```
-- Table for Genre entity with its attributes
CREATE TABLE Genre (
    Genre_ID INTEGER PRIMARY KEY,
    Genre_Name TEXT NOT NULL,
    Description TEXT
);
```

Inserted Genre Table Data

```
--Insert Into Genre
INSERT INTO Genre (Genre_ID, Genre_Name, Description)
VALUES
(1, 'General Fiction', 'Literary works with a focus on character and plot development, exploring various themes and human experiences.'),
(2, 'Mystery & Thriller', 'Suspenseful stories centered around crime, investigation, or espionage with an emphasis on tension and excitement.'),
(3, 'Science Fiction & Fantasy', 'Imaginative works that explore alternate realities, futuristic concepts, and magical or supernatural elements.'),
(4, 'Horror & Suspense', 'Stories designed to evoke fear, unease, or dread, often featuring supernatural or psychological elements.'),
(5, 'Dystopian & Apocalyptic', 'Depictions of societies in decline or collapse, often exploring themes of political and social oppression or environmental disaster.'),
(6, 'Classics', 'Enduring works of literature that have stood the test of time, often featuring rich language and complex themes.'),
(7, 'Historical Fiction', 'Fictional stories set in the past, often based on real historical events or figures, and exploring the customs and experiences of that time.'),
(8, 'Epic Poetry & Mythology', 'Ancient or traditional stories and poems, often featuring heroes, gods, and mythical creatures, and exploring cultural values and beliefs.');

SELECT * FROM Genre;
```

Output:

	genre_id [PK] integer	genre_name text	description text
1	1	General Fiction	Literary works with a focus on character and plot development, exploring various themes and human experiences.
2	2	Mystery & Thriller	Suspenseful stories centered around crime, investigation, or espionage with an emphasis on tension and excitement.
3	3	Science Fiction & Fantasy	Imaginative works that explore alternate realities, futuristic concepts, and magical or supernatural elements.
4	4	Horror & Suspense	Stories designed to evoke fear, unease, or dread, often featuring supernatural or psychological elements.
5	5	Dystopian & Apocalyptic	Depictions of societies in decline or collapse, often exploring themes of political and social oppression or environmental disaster.
6	6	Classics	Enduring works of literature that have stood the test of time, often featuring rich language and complex themes.
7	7	Historical Fiction	Fictional stories set in the past, often based on real historical events or figures, and exploring the customs and experiences of that t...
8	8	Epic Poetry & Mythology	Ancient or traditional stories and poems, often featuring heroes, gods, and mythical creatures, and exploring cultural values and beli...

Created Catalog Table

```
-- Table for Catalog entity with its attributes
CREATE TABLE Catalog (
    Catalog_ID INTEGER PRIMARY KEY,
    Catalog_Name TEXT NOT NULL,
    Location TEXT NOT NULL
);
```

Inserted Catalog Table Data

```
--Insert Into Catalog
INSERT INTO Catalog (Catalog_ID, Catalog_Name, Location)
VALUES (1, 'Books', 'A1.1'),
       (2, 'Magazines', 'B2.1'),
       (3, 'E-Books', 'C3.1'),
       (4, 'Audiobooks', 'D4.1'),
       (5, 'Journals', 'E5.1'),
       (6, 'Newspaper', 'F6.1'),
       (7, 'Maps', 'G7.1'),
       (8, 'Novels', 'H8.1'),
       (9, 'Sheet Music', 'I9.1'),
       (10, 'Educational', 'J10.1');

SELECT * FROM Catalog;
```

Output

	catalog_id [PK] integer	catalog_name text	location text
1	1	Books	A1.1
2	2	Magazines	B2.1
3	3	E-Books	C3.1
4	4	Audiobooks	D4.1
5	5	Journals	E5.1
6	6	Newspaper	F6.1
7	7	Maps	G7.1
8	8	Novels	H8.1
9	9	Sheet Music	I9.1
10	10	Educational	J10.1

Created Material Table

```
-- Table for Material entity with its attributes
CREATE TABLE Material (
    Material_ID INTEGER PRIMARY KEY,
    Title TEXT NOT NULL,
    Publication_Date DATE,
    Catalog_ID INTEGER NOT NULL,
    Genre_ID INTEGER NOT NULL,
    CONSTRAINT FK_Catalog_Material FOREIGN KEY (Catalog_ID)
        REFERENCES Catalog(Catalog_ID) ON DELETE CASCADE,
    CONSTRAINT FK_Genre_Material FOREIGN KEY (Genre_ID)
        REFERENCES Genre(Genre_ID) ON DELETE CASCADE
);
```

Inserted Material Table Data

```sql
--Insert Into Material
INSERT INTO Material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID)
VALUES
    (1, 'The Catcher in the Rye', '1951-07-16', 1, 1),
    (2, 'To Kill a Mockingbird', '1960-07-11', 2, 1),
    (3, 'The Da Vinci Code', '2003-04-01', 3, 2),
    (4, 'The Hobbit', '1937-09-21', 4, 3),
    (5, 'The Shining', '1977-01-28', 5, 4),
    (6, 'Pride and Prejudice', '1813-01-28', 1, 1),
    (7, 'The Great Gatsby', '1925-04-10', 2, 1),
    (8, 'Moby Dick', '1851-10-18', 3, 1),
    (9, 'Crime and Punishment', '1866-01-01', 4, 1),
    (10, 'The Hitchhiker''s Guide to the Galaxy', '1979-10-12', 5, 3),
    (11, '1984', '1949-06-08', 1, 5),
    (12, 'Animal Farm', '1945-08-17', 2, 5),
    (13, 'The Haunting of Hill House', '1959-10-17', 3, 4),
    (14, 'Brave New World', '1932-08-01', 4, 5),
    (15, 'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe', '1950-10-16', 5, 3),
    (16, 'The Adventures of Huckleberry Finn', '1884-12-10', 6, 1),
    (17, 'Catch-22', '1961-10-11', 7, 1),
    (18, 'The Picture of Dorian Gray', '1890-07-01', 8, 1),
    (19, 'The Call of Cthulhu', '1928-02-01', 9, 4),
    (20, 'Harry Potter and the Philosopher''s Stone', '1997-06-26', 10, 3),
    (21, 'Frankenstein', '1818-01-01', 6, 4),
    (22, 'A Tale of Two Cities', '1859-04-30', 7, 1),
    (23, 'The Iliad', '1750-01-01', 8, 6),
    (24, 'The Odyssey', '1725-01-01', 9, 6),
    (25, 'The Brothers Karamazov', '1880-01-01', 10, 1),
    (26, 'The Divine Comedy', '1320-01-01', 6, 6),
    (27, 'The Grapes of Wrath', '1939-04-14', 7, 1),
    (28, 'The Old Man and the Sea', '1952-09-01', 8, 1),
    (29, 'The Count of Monte Cristo', '1844-01-01', 9,1),
    (30, 'A Midsummer Nights Dream','1596-01-01',10,7),
    (31, 'The Tricky Book','1888-01-01',10,7);

SELECT * FROM Material;
```

Output

Data Output Messages Notifications

	material_id [PK] integer	title text	publication_date date	catalog_id integer	genre_id integer
1	1	The Catcher in the Rye	1951-07-16	1	1
2	2	To Kill a Mockingbird	1960-07-11	2	1
3	3	The Da Vinci Code	2003-04-01	3	2
4	4	The Hobbit	1937-09-21	4	3
5	5	The Shining	1977-01-28	5	4
6	6	Pride and Prejudice	1813-01-28	1	1
7	7	The Great Gatsby	1925-04-10	2	1
8	8	Moby Dick	1851-10-18	3	1
9	9	Crime and Punishment	1866-01-01	4	1
10	10	The Hitchhiker's Guide to the Galaxy	1979-10-12	5	3
11	11	1984	1949-06-08	1	5
12	12	Animal Farm	1945-08-17	2	5
13	13	The Haunting of Hill House	1959-10-17	3	4
14	14	Brave New World	1932-08-01	4	5
15	15	The Chronicles of Narnia: The Lion, the Witch and the Wardro…	1950-10-16	5	3
16	16	The Adventures of Huckleberry Finn	1884-12-10	6	1
17	17	Catch-22	1961-10-11	7	1
18	18	The Picture of Dorian Gray	1890-07-01	8	1
19	19	The Call of Cthulhu	1928-02-01	9	4
20	20	Harry Potter and the Philosopher's Stone	1997-06-26	10	3
21	21	Frankenstein	1818-01-01	6	4
22	22	A Tale of Two Cities	1859-04-30	7	1
23	23	The Iliad	1750-01-01	8	6
24	24	The Odyssey	1725-01-01	9	6
25	25	The Brothers Karamazov	1880-01-01	10	1
26	26	The Divine Comedy	1320-01-01	6	6
27	27	The Grapes of Wrath	1939-04-14	7	1
28	28	The Old Man and the Sea	1952-09-01	8	1
29	29	The Count of Monte Cristo	1844-01-01	9	1
30	30	A Midsummer Nights Dream	1596-01-01	10	7
31	31	The Tricky Book	1888-01-01	10	7
32	32	New book	2020-08-01	3	2

Created Borrow Table

```
-- Table for Borrow entity with its attributes
CREATE TABLE Borrow (
    Borrow_ID INTEGER PRIMARY KEY,
    Material_ID INTEGER NOT NULL,
    Member_ID INTEGER NOT NULL,
    Staff_ID INTEGER NOT NULL,
    Borrow_Date DATE NOT NULL,
    Due_Date DATE NOT NULL,
    Return_Date DATE,
    CONSTRAINT FK_Material_Borrow FOREIGN KEY (Material_ID)
        REFERENCES Material(Material_ID) ON DELETE CASCADE,
    CONSTRAINT FK_Member_Borrow FOREIGN KEY (Member_ID)
        REFERENCES Member(Member_ID) ON DELETE CASCADE,
    CONSTRAINT FK_Staff_Borrow FOREIGN KEY (Staff_ID)
        REFERENCES Staff(Staff_ID) ON DELETE CASCADE
);
```

Inserted Borrow Table Data

```
--Insert Into Borrow
INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)
VALUES
(1, 1, 1, 1, '2018-09-12', '2018-10-03', '2018-09-30'),
(2, 2, 2, 1, '2018-10-15', '2018-11-05', '2018-10-29'),
(3, 3, 3, 1, '2018-12-20', '2019-01-10', '2019-01-08'),
(4, 4, 4, 1, '2019-03-11', '2019-04-01', '2019-03-27'),
(5, 5, 5, 1, '2019-04-20', '2019-05-11', '2019-05-05'),
(6, 6, 6, 1, '2019-07-05', '2019-07-26', '2019-07-21'),
(7, 7, 7, 1, '2019-09-10', '2019-10-01', '2019-09-25'),
(8, 8, 8, 1, '2019-11-08', '2019-11-29', '2019-11-20'),
(9, 9, 9, 1, '2020-01-15', '2020-02-05', '2020-02-03'),
(10, 10, 10, 1, '2020-03-12', '2020-04-02', '2020-03-28'),
(11, 1, 11, 2, '2020-05-14', '2020-06-04', '2020-05-28'),
(12, 2, 12, 2, '2020-07-21', '2020-08-11', '2020-08-02'),
(13, 3, 13, 2, '2020-09-25', '2020-10-16', '2020-10-15'),
(14, 4, 1, 2, '2020-11-08', '2020-11-29', '2020-11-24'),
(15, 5, 2, 2, '2021-01-03', '2021-01-24', '2021-01-19'),
(16, 6, 3, 2, '2021-02-18', '2021-03-11', '2021-03-12'),
(17, 17, 4, 2, '2021-04-27', '2021-05-18', '2021-05-20'),
(18, 18, 5, 2, '2021-06-13', '2021-07-04', '2021-06-28'),
(19, 19, 6, 2, '2021-08-15', '2021-09-05', '2021-09-03'),
(20, 20 , 7, 2, '2021-10-21', '2021-11-11', '2021-11-05'),
(21 , 21 , 1 , 3 , '2021-11-29' , '2021-12-20', NULL),
(22 , 22 , 2 , 3 , '2022-01-10' , '2022-01-31' , '2022-01-25'),
(23 , 23 , 3 , 3 , '2022-02-07' , '2022-02-28' , '2022-02-23'),
(24 , 24 , 4 , 3 , '2022-03-11' , '2022-04-01' , '2022-03-28'),
(25 , 25 , 5 , 3 , '2022-04-28' , '2022-05-19' , '2022-05-18'),
(26 , 26 , 6 , 3 , '2022-06-22' , '2022-07-13' , '2022-07-08'),
(27 , 27 , 7 , 3 , '2022-08-04' , '2022-08-25' , '2022-08-23'),
(28 , 28 , 8 , 3 , '2022-09-13' , '2022-10-04' , '2022-09-28'),
(29 , 29 , 9 , 3 , '2022-10-16' , '2022-11-06' , '2022-11-05'),
(30 , 30 , 8 , 3 , '2022-11-21' , '2022-12-12' , '2022-12-05'),
(31 , 1 , 9 , 4 , '2022-12-28' , '2023-01-18' , NULL),
(32 , 2 , 1 , 4 , '2023-01-23' , '2023-02-13', NULL ),
(33 , 3 , 10 , 4 , '2023-02-02' , '2023-02-23' , '2023-02-17'),
(34 , 4 , 11 , 4 , '2023-03-01' , '2023-03-22', NULL ),
(35 , 5 , 12 , 4 , '2023-03-10' , '2023-03-31', NULL),
(36 , 6 , 13 , 4 , '2023-03-15' , '2023-04-05', NULL  ),
(37 , 7 , 17 , 4 , '2023-03-25' , '2023-04-15', NULL ),
(38 , 8 , 8 , 4 , '2023-03-30' , '2023-04-20', NULL ),
(39 , 9 , 9 , 4 , '2023-03-26' , '2023-04-16', NULL ),
(40, 10, 20, 4,'2023-03-28', '2023-04-18', NULL);
```

Output

Data Output | Messages | Notifications

	borrow_id [PK] integer	material_id integer	member_id integer	staff_id integer	borrow_date date	due_date date	return_date date
1	1	1	1	1	2018-09-12	2018-10-03	2018-09-30
2	2	2	2	1	2018-10-15	2018-11-05	2018-10-29
3	3	3	3	1	2018-12-20	2019-01-10	2019-01-08
4	4	4	4	1	2019-03-11	2019-04-01	2019-03-27
5	6	6	6	1	2019-07-05	2019-07-26	2019-07-21
6	7	7	7	1	2019-09-10	2019-10-01	2019-09-25
7	8	8	8	1	2019-11-08	2019-11-29	2019-11-20
8	9	9	9	1	2020-01-15	2020-02-05	2020-02-03
9	10	10	10	1	2020-03-12	2020-04-02	2020-03-28
10	11	1	11	2	2020-05-14	2020-06-04	2020-05-28
11	12	2	12	2	2020-07-21	2020-08-11	2020-08-02
12	13	3	13	2	2020-09-25	2020-10-16	2020-10-15
13	14	4	1	2	2020-11-08	2020-11-29	2020-11-24
14	15	5	2	2	2021-01-03	2021-01-24	2021-01-19
15	16	6	3	2	2021-02-18	2021-03-11	2021-03-12
16	17	17	4	2	2021-04-27	2021-05-18	2021-05-20
17	19	19	6	2	2021-08-15	2021-09-05	2021-09-03
18	20	20	7	2	2021-10-21	2021-11-11	2023-04-01
19	21	21	1	3	2021-11-29	2021-12-20	[null]
20	22	22	2	3	2022-01-10	2022-01-31	2022-01-25
21	23	23	3	3	2022-02-07	2022-02-28	2022-02-23
22	24	24	4	3	2022-03-11	2022-04-01	2022-03-28
23	26	26	6	3	2022-06-22	2022-07-13	2022-07-08
24	27	27	7	3	2022-08-04	2022-08-25	2022-08-23
25	28	28	8	3	2022-09-13	2022-10-04	2022-09-28
26	29	29	9	3	2022-10-16	2022-11-06	2022-11-05
27	30	30	8	3	2022-11-21	2022-12-12	2022-12-05
28	31	1	9	4	2022-12-28	2023-01-18	[null]
29	32	2	1	4	2023-01-23	2023-02-13	[null]
30	33	3	10	4	2023-02-02	2023-02-23	2023-02-17
31	34	4	11	4	2023-03-01	2023-03-22	[null]
32	35	5	12	4	2023-03-10	2023-03-31	[null]
33	36	6	13	4	2023-03-15	2023-04-05	[null]
34	37	7	17	4	2023-03-25	2023-04-15	[null]
35	38	8	8	4	2023-03-30	2023-04-20	[null]
36	39	9	9	4	2023-03-26	2023-04-16	[null]
37	40	10	20	4	2023-03-28	2023-04-18	[null]

Created Aurthorship Table

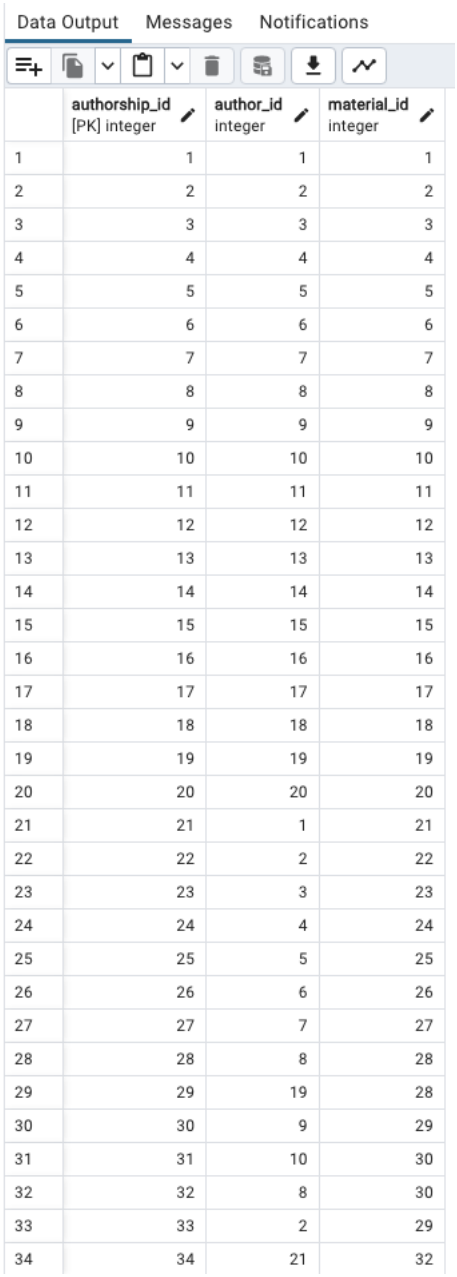
```sql
-- Table for Authorship entity with its attributes
CREATE TABLE Authorship (
    Authorship_ID INTEGER PRIMARY KEY,
    Author_ID INTEGER NOT NULL,
    Material_ID INTEGER NOT NULL,
    CONSTRAINT FK_Author_Authorship FOREIGN KEY (Author_ID)
        REFERENCES Author(Author_ID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT FK_Material_Authorship FOREIGN KEY (Material_ID)
        REFERENCES Material(Material_ID) ON DELETE CASCADE
);
```

Inserted Aurthorship Table Data

```sql
--Insert Into Authorship
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
VALUES
(1, 1, 1),
(2, 2, 2),
(3, 3, 3),
(4, 4, 4),
(5, 5, 5),
(6, 6, 6),
(7, 7, 7),
(8, 8, 8),
(9, 9, 9),
(10, 10, 10),
(11, 11, 11),
(12, 12, 12),
(13, 13, 13),
(14, 14, 14),
(15, 15, 15),
(16, 16, 16),
(17, 17, 17),
(18, 18, 18),
(19, 19, 19),
(20, 20, 20),
(21, 1, 21),
(22, 2, 22),
(23, 3, 23),
(24, 4, 24),
(25, 5, 25),
(26, 6, 26),
(27, 7, 27),
(28, 8, 28),
(29, 19, 28),
(30, 9, 29),
(31, 10, 30),
(32, 8, 30),
(33, 2, 29);

SELECT * FROM Authorship;
```

Output



Data Output | Messages | Notifications

	authorship_id [PK] integer	author_id integer	material_id integer
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11
12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19
20	20	20	20
21	21	1	21
22	22	2	22
23	23	3	23
24	24	4	24
25	25	5	25
26	26	6	26
27	27	7	27
28	28	8	28
29	29	19	28
30	30	9	29
31	31	10	30
32	32	8	30
33	33	2	29
34	34	21	32

Querying/Updates:

1. Which materials are currently available in the library?

```
--1. Which materials are currently available in the library? If a material is bo

SELECT m.Title, m.Material_ID
FROM Material m
LEFT JOIN Borrow b ON m.Material_ID = b.Material_ID AND b.Return_Date IS NULL
WHERE b.Material_ID IS NULL;
```

Output:

Data Output | Messages | Notifications

	title text	material_id [PK] integer
1	The Da Vinci Code	3
2	1984	11
3	Animal Farm	12
4	The Haunting of Hill House	13
5	Brave New World	14
6	The Chronicles of Narnia: The Lion, the Witch and the Wardro...	15
7	The Adventures of Huckleberry Finn	16
8	Catch-22	17
9	The Picture of Dorian Gray	18
10	The Call of Cthulhu	19
11	Harry Potter and the Philosopher's Stone	20
12	A Tale of Two Cities	22
13	The Iliad	23
14	The Odyssey	24
15	The Brothers Karamazov	25
16	The Divine Comedy	26
17	The Grapes of Wrath	27
18	The Old Man and the Sea	28
19	The Count of Monte Cristo	29
20	A Midsummer Nights Dream	30
21	The Tricky Book	31
22	New book	32

This SQL code selects materials from the "Material" table that are available for borrowing. It does this by checking for materials with no active borrow records (where "Return_Date" is NULL) in the "Borrow" table and then retrieves the "Title" and "Material_ID" for these materials.

2. Which materials are currently overdue?
Suppose today is 04/01/2023, and show the borrow date and due date of each material

```
--2. Which materials are currently overdue? Suppose today is 04

SELECT m.Title, b.Borrow_Date, b.Due_Date
FROM Material m
INNER JOIN Borrow b ON m.Material_ID = b.Material_ID
WHERE b.Return_Date IS NULL AND b.Due_Date < '2023-04-01';
```

Output:

	title text	borrow_date date	due_date date
1	Frankenstein	2021-11-29	2021-12-20
2	The Catcher in the Rye	2022-12-28	2023-01-18
3	To Kill a Mockingbird	2023-01-23	2023-02-13
4	The Hobbit	2023-03-01	2023-03-22
5	The Shining	2023-03-10	2023-03-31

This SQL code selects the "Title," "Borrow_Date," and "Due_Date" for materials that have been borrowed but not returned, and their "Due_Date" is before April 1, 2023. It retrieves information about materials that are currently borrowed and overdue as of that date.

3. What are the top 10 most borrowed materials in the library?
Show the title of each material and order them based on their available counts

```
SELECT m.Title,
       COUNT(*) AS Borrow_Count,
       SUM(CASE WHEN b.Return_Date IS NULL THEN 1 ELSE 0 END) AS Available_Count
FROM Material m
JOIN Borrow b ON m.Material_ID = b.Material_ID
GROUP BY m.Title
ORDER BY Available_Count DESC
LIMIT 10;
```

Output:

	title text	borrow_count bigint	available_count bigint
1	The Hobbit	3	1
2	The Catcher in the Rye	3	1
3	Frankenstein	1	1
4	The Hitchhiker's Guide to the Galaxy	2	1
5	Moby Dick	2	1
6	To Kill a Mockingbird	3	1
7	The Shining	2	1
8	Crime and Punishment	2	1
9	Pride and Prejudice	3	1
10	The Great Gatsby	2	1

This SQL code retrieves the top 10 materials with the most borrow activity. It lists the material titles, the total number of times they have been borrowed (Borrow_Count), and how many of them are currently available for borrowing (Available_Count). The materials are grouped by title and displayed in descending order of availability.

4. How many books has the author Lucas Piki written?

```
--4. How many materials has the author Lucas Piki written?

SELECT m.Title
FROM Material m
JOIN Authorship a ON m.Material_ID = a.Material_ID
JOIN Author au ON a.Author_ID = au.Author_ID
WHERE au.Author_Name = 'Lucas Piki';
```

Output:

Data Output | Messages | Notifications

	title text
1	Harry Potter and the Philosopher's Stone

This SQL code retrieves the titles of materials authored by 'Lucas Piki'. It does this by joining the "Material," "Authorship," and "Author" tables. The "Authorship" table links materials to authors, and the "Author" table provides author information. The query filters the results to include only materials where the author's name is 'Lucas Piki.'

5. How many books were written by two or more authors?

```
--5. How many materials were written by two or more authors?

SELECT COUNT(m.Material_ID) AS MaterialsWithMultipleAuthors
FROM Material m
JOIN Authorship a ON m.Material_ID = a.Material_ID
GROUP BY m.Material_ID
HAVING COUNT(DISTINCT a.Author_ID) >= 2;
```

Output:

	materialswithmultipleauthors bigint
1	2
2	2
3	2

This SQL code counts the number of materials that have multiple authors. It does so by joining the "Material" and "Authorship" tables and then grouping the results by the "Material_ID." The `HAVING` clause filters the grouped results to include only materials that have at least two distinct author IDs associated with them. The count of such materials is then labeled as "MaterialsWithMultipleAuthors."

6. What are the most popular genres in the library?

```
SELECT g.Genre_Name, COUNT(b.Material_ID) AS borrow_Count
FROM Genre g
LEFT JOIN Material m ON g.Genre_ID = m.Genre_ID
LEFT JOIN Borrow b ON m.Material_ID = b.Material_ID
GROUP BY g.Genre_Name
ORDER BY borrow_Count DESC;
```

Output:

	genre_name text	borrow_count bigint
1	General Fiction	20
2	Science Fiction & Fantasy	6
3	Horror & Suspense	4
4	Mystery & Thriller	3
5	Classics	3
6	Historical Fiction	1
7	Dystopian & Apocalyptic	0
8	Epic Poetry & Mythology	0

This SQL code counts the number of materials borrowed for each genre, displaying the genre name and the corresponding borrow count. It lists genres in descending order of borrow counts, showing the most popular genres at the top.

7. How many materials have been borrowed from 09/2020-10/2020?

```
SELECT COUNT(DISTINCT Material_ID) as Borrowed_Materials
FROM Borrow
WHERE Borrow_Date >= '2020-09-01' AND Borrow_Date <= '2020-10-31';
```

Output:

	borrowed_materials bigint
1	1

This SQL code counts the number of distinct materials that were borrowed within the date range of September 1, 2020, to October 31, 2020. The result is labeled as "Borrowed_Materials," representing the count of unique materials borrowed during that specific period.

8. How do you update the “Harry Potter and the Philosopher's Stone” when it is returned on 04/01/2023?

```
UPDATE Borrow
SET Return_Date = '2023-04-01'
WHERE Material_ID IN (
  SELECT Material_ID
  FROM Material
  WHERE Title = 'Harry Potter and the Philosopher''s Stone'
);
```

This SQL code performs an update operation on the "Borrow" table. It sets the "Return_Date" to '2023-04-01' for all records in the "Borrow" table where the associated "Material_ID" matches the "Material_ID" of the material with the title 'Harry Potter and the Philosopher's Stone' in the "Material" table. In other words, it updates the return date for all instances of the book 'Harry Potter and the Philosopher's Stone' to the specified date.

9. How do you delete the member Emily Miller and all her related records from the database?

```
-- First delete related records in the Borrow table
DELETE FROM Borrow
WHERE Member_ID = (SELECT Member_ID FROM Member WHERE Member_Name = 'Emily Miller');

-- Then delete the member from the Member table
DELETE FROM Member
WHERE Member_Name = 'Emily Miller';
```

This SQL code performs two operations:

1. It first deletes all records from the "Borrow" table where the "Member_ID" matches the "Member_ID" of the member with the name 'Emily Miller' in the "Member" table. This removes all borrow records associated with the member 'Emily Miller.'

2. After deleting the associated borrow records, it proceeds to delete the member 'Emily Miller' from the "Member" table.

These two SQL statements are used to remove both the borrow history and the member from the database.

10. How do you add the following material to the database? Title: New book Date: 2020-08-01 Catalog: E-Books Genre: Mystery & Thriller Author: Lucas Pipi

```
--10. How do you add the following material to the database?
--Title: New book
--Date: 2020-08-01
--Catalog: E-Books
--Genre: Mystery & Thriller
--Author: Lucas Luke

INSERT INTO material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID)
VALUES (32, 'New book', '2020-08-01',
        (SELECT Catalog_ID FROM Catalog WHERE Catalog_Name = 'E-Books' LIMIT 1),
        (SELECT Genre_ID FROM Genre WHERE Genre_Name = 'Mystery & Thriller' LIMIT 1));


-- Add the new author
INSERT INTO Author (Author_ID, Author_Name, Birth_Date, Nationality)
VALUES (21, 'Lucas Pipi', NULL, NULL);

-- Add the authorship record linking the new material to the new author
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
VALUES (34, 21, 32);
```

These SQL statements add a new book called 'New book' to the database, associated with the 'E-Books' catalog and the 'Mystery & Thriller' genre. They also add a new author named 'Lucas Pipi' to the database and establish the authorship link between the new book and the new author.

Design You are required to explain how to extend the existing database system to incorporate the following features. While you can provide SQL statements to illustrate your concepts, you don't need to execute the statements.

1. Alert staff about overdue materials on a daily-basis?

I can implement a scheduled job or a script that runs daily and checks for overdue materials. If any materials are overdue, it can send alerts to the staff. This can be achieved using a combination of SQL and a scripting language like Python or a job scheduling tool. Here's a high-level outline of the process:

Create a script that connects to the database.
Query the Borrow table to identify materials that are overdue based on the Due_Date.
If overdue materials are found, send notifications or alerts to the staff using email, SMS, or any other preferred communication methods

2. Automatically deactivate the membership based on the member’s overdue occurrence (>= three times). And reactivate the membership once the member pays the overdue fee

I will need to add additional functionality to manage the member's status and track overdue occurrences. Here's a high-level outline of the process:

-Add a new column to the Member table to track the number of overdue occurrences for each member.
-Create a script or database trigger that runs whenever a member borrows or returns a material. This script should check the member's borrowing history and update the overdue occurrence count.
-When a member exceeds three overdue occurrences, update the member's status to "Deactivated."
-Create a process that allows the member to pay the overdue fees. Upon payment, update the overdue occurrence count and reactivate the membership.
-I can also create a scheduled job or trigger that checks for overdue materials daily and increments the overdue occurrence count for members with overdue materials.
-When the overdue occurrence count reaches three or more, the member's status is set to "Deactivated."