CS-504

Principles of Data Management and Mining

Design and Implementation of Database Management System for a Public Library

Professor Binqian Yin

Junaid Mohammed (G01386670)

Database Design:

1. Scope of the Project:

Designing a database schema for a library management system that keeps tracks the library's collection of materials, their catalogs, the genres to which they belong to, borrowing activity of members in the library, the authors of the materials, and the staff who manage the library.

Business rules that describe all entities, relationships constraints:

1. Material Entity:
   A Material can be associated with a single Catalog entry and a single Genre.
   An Author can author multiple Materials.
   Attributes: Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID.

2. Catalog:
   A Catalog can have multiple Materials.
   Attributes: Catalog_ID, Name, Location.

3. Genre:
   A Genre can be associated with multiple Materials.
   Attributes: Genre_ID, Name, Description

4. Borrow:
   A Borrow transaction can involve a single Material and a single Member.
   A Borrow transaction is processed by a single Staff member.
   Attributes: Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date.

5. Author:
   An Author can author multiple Materials.
   Attributes: Author_ID, Name, Birth_Date, Nationality.

6. Authorship:
   An Authorship record associates a single Author with a single Material.
   Attributes: Authorship_ID, Author_ID, Material_ID

7. Member:
   A Member can borrow multiple Materials.
   Attributes: Member_ID, Name, Contact_Info, Join_Date

8. Staff:
   A Staff member can process multiple Borrow transactions.
   Attributes: Staff_ID, Name, Contact_Info, Job_Title, Hire_Date.

Constraints:

I. Primary key:

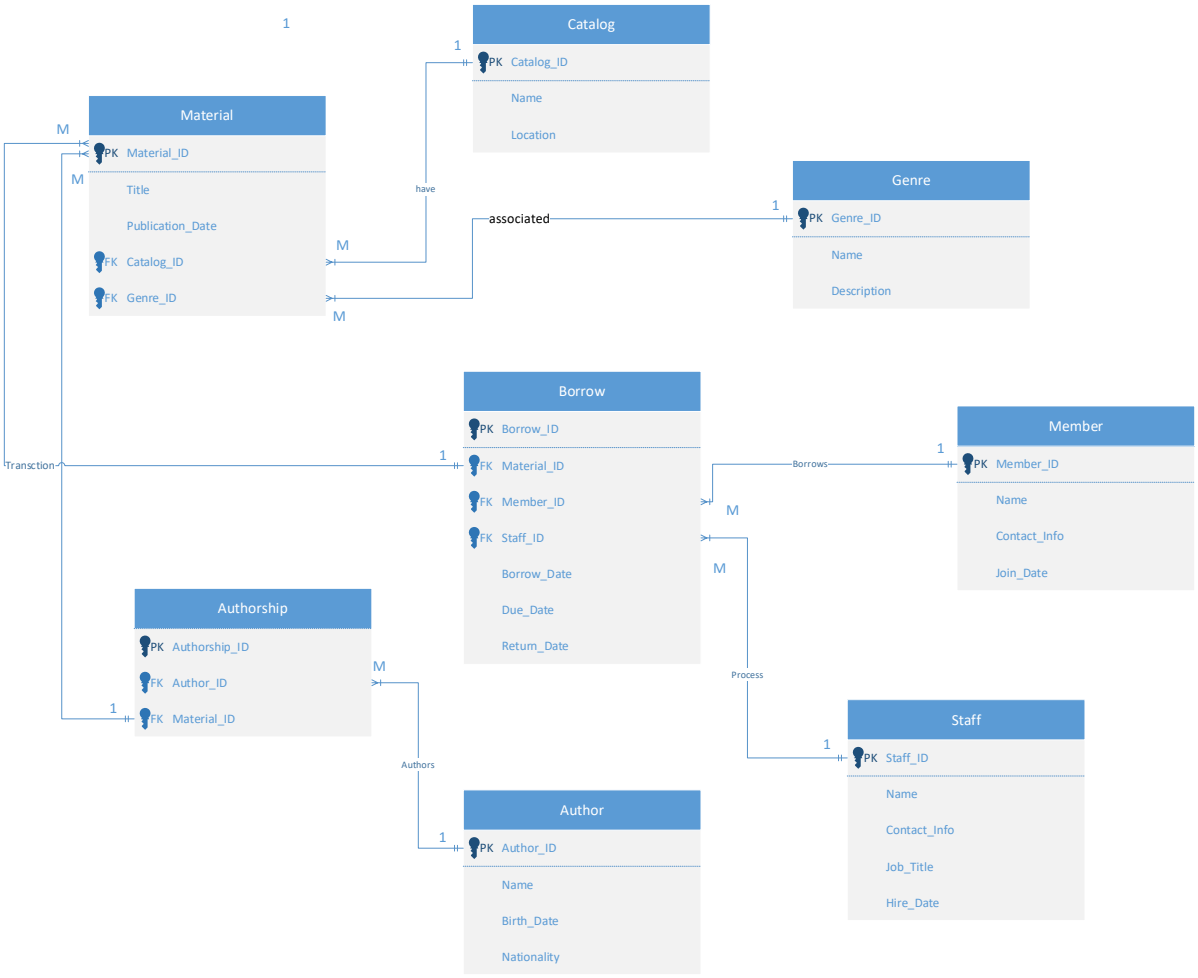
- Material_ID, Borrow_ID, Genre_ID, Catalog_ID, Author_ID, Authorship_ID, Member_ID, and Staff_ID should all be unique identifiers for their respective tables.

II. Foreign key:
- Material_ID & Member_ID in the Borrow table should reference the Material and Member tables respectively.
- Material_ID in the Authorship table should reference the Material table.
- Author_ID in the Authorship table should reference the Author table.
- Catalog_ID and Genre_ID in the Material table should reference the Catalog and Genre tables respectively.

III. Attribute:
- Due_Date in the Borrow table should be after the Borrow_Date.
- Publication_Date in the Material table should not be after the current date.
- Join_Date in the Member table should be before the current date.

Relationships Explained:
- A Material has many Authorship entries (1:M relationship).
- A Material has one Genre (M:1 relationship).
- A Material has one Catalog entry (M:1 relationship).
- An Author has many Authorship entries (1:M relationship).
- An Authorship entry has one Material and one Author (many-to-one-to-many relationship).
- A Member has many Borrow records (1:M relationship).
- A Catalog entry has one Material (1:M relationship).
- A Borrow field has one Material, one Member, and one Staff (many-to-one-to-many relationship).

2. **Entity-Relationship (ER) diagram:**
   - ❖ I have created an Entity-Relationship (ER) diagram using Microsoft Visio that represents the database schema for a public library system.



Database Implementation:

1. I've chosen **Postgre** Database Management System (DBMS) for this project.
2. **Implementing** DBMS's data definition language (DDL) for each table with the given data.

- On given data creating all the 8 tables in the Postgre

  Database: CS Final Project

```
CREATE TABLE Catalog (
   Catalog_ID INT PRIMARY KEY,
   Name VARCHAR(180),
   Location VARCHAR(150)
);

CREATE TABLE Genre (
```

```
    Genre_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Description TEXT
);

CREATE TABLE Material (
    Material_ID INT PRIMARY KEY,
    Title VARCHAR(250),
    Publication_Date DATE,
    Catalog_ID INT,
    Genre_ID INT,
    FOREIGN KEY (Catalog_ID) REFERENCES Catalog (Catalog_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre (Genre_ID),
            CHECK (Publication_Date <= CURRENT_DATE)
);

CREATE TABLE Member (
    Member_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Contact_Info VARCHAR(200),
    Join_Date DATE,
            CHECK (Join_Date <= CURRENT_DATE)
);


CREATE TABLE Staff (
    Staff_ID INT PRIMARY KEY,
    Name VARCHAR(400),
    Contact_Info VARCHAR(200),
    Job_Title VARCHAR(255),
    Hire_Date DATE,
            CHECK (Hire_Date <= CURRENT_DATE)
);


CREATE TABLE Borrow (
    Borrow_ID INT PRIMARY KEY,
    Material_ID INT,
    Member_ID INT,
    Staff_ID INT,
    Borrow_Date DATE,
    Due_Date DATE,
    Return_Date DATE,
    FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID),
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
    FOREIGN KEY (Staff_ID) REFERENCES Staff(Staff_ID),
            CHECK (Borrow_Date <= CURRENT_DATE AND Due_Date >= Borrow_Date AND
Return_Date >= Borrow_Date)
```

```
);


CREATE TABLE Author (
   Author_ID INT PRIMARY KEY,
   Name VARCHAR(400),
   Birth_Date DATE,
   Nationality VARCHAR(100),
         CHECK (Birth_Date <= CURRENT_DATE)
);


CREATE TABLE Authorship (
   Authorship_ID INT PRIMARY KEY,
   Author_ID INT,
   Material_ID INT,
   FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID),
   FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID)
);
```

3. **Populating the sample data into database**.

Inserting Values into the Tables:

Here after creating tables, I've populated the sample data in Postgre and pasted the screenshot of each table creation with its outputs.

Catalog:

Query Query History

```
/*Inserting_Values_into_the_Tables*/
INSERT INTO Catalog (Catalog_ID, 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;
```

Data Output Messages Notifications

	catalog_id [PK] integer	name character varying (180)	location character varying (150)
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

Total rows: 10 of 10 Query complete 00:00:00.056 Ln 106, Col 1

Genre:

Query Query History

```
107
108 INSERT INTO Genre (Genre_ID, Name, Description)
109 VALUES (1, 'General Fiction', 'Literary works with a focus on character and plot development, exploring various themes an
110 (2, 'Mystery & Thriller', 'Suspenseful stories centered around crime, investigation, or espionage with an emphasis on ten
111 (3, 'Science Fiction & Fantasy', 'Imaginative works that explore alternate realities, futuristic concepts, and magical or
112 (4, 'Horror & Suspense', 'Stories designed to evoke fear, unease, or dread, often featuring supernatural or psychological
113 (5, 'Dystopian & Apocalyptic', 'Depictions of societies in decline or collapse, often exploring themes of political and s
114 (6, 'Classics', 'Enduring works of literature that have stood the test of time, often featuring rich language and complex
115 (7, 'Historical Fiction', 'Fictional stories set in the past, often based on real historical events or figures, and explo
116 (8, 'Epic Poetry & Mythology', 'Ancient or traditional stories and poems, often featuring heroes, gods, and mythical crea
117 SELECT * FROM Genre;
118
```

Data Output Messages Notifications

	genre_id [PK] integer	name character varying (400)	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...

Total rows: 8 of 8 Query complete 00:00:00.058 Ln 118, Col 1

Material:

Query Query History

```
120 INSERT INTO Material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID)
121 VALUES (1, 'The Catcher in the Rye', '1951-07-16', 1, 1),
122 (2, 'To Kill a Mockingbird', '1960-07-11', 2, 1),
123 (3, 'The Da Vinci Code', '2003-04-01', 3, 2),
124 (4, 'The Hobbit', '1937-09-21', 4, 3),
125 (5, 'The Shining', '1977-01-28', 5, 4),
126 (6, 'Pride and Prejudice', '1813-01-28', 1, 1),
127 (7, 'The Great Gatsby', '1925-04-10', 2, 1),
128 (8, 'Moby Dick', '1851-10-18', 3, 1),
129 (9, 'Crime and Punishment', '1866-01-01', 4, 1),
130 (10, 'The Hitchhiker''s Guide to the Galaxy', '1979-10-12', 5, 3),
131 (11, '1984', '1949-06-08', 1, 5),
132 (12, 'Animal Farm', '1945-08-17', 2, 5),
133 (13, 'The Haunting of Hill House', '1959-10-17', 3, 4),
134 (14, 'Brave New World', '1932-08-01', 4, 5),
135 (15, 'The Chronicles of Narnia: The Lion, the Witch and the Wardrobe', '1950-10-16', 5, 3),
136 (16, 'The Adventures of Huckleberry Finn', '1884-12-10', 6, 1),
137 (17, 'The Catch-22', '1961-10-11', 7, 1),
138 (18, 'The Picture of Dorian Gray', '1890-07-01', 8, 1),
139 (19, 'The Call of Cthulhu', '1928-02-01', 9, 4),
140 (20, 'Harry Potter and the Philosopher''s Stone', '1997-06-26', 10, 3),
141 (21, 'Frankenstein', '1818-01-01', 6, 4),
142 (22, 'A Tale of Two Cities', '1859-04-30', 7, 1),
143 (23, 'The Iliad', '1750-01-01', 8, 6),
144 (24, 'The Odyssey', '1725-01-01', 9, 6),
145 (25, 'The Brothers Karamazov', '1880-01-01', 10, 1),
146 (26, 'The Divine Comedy', '1320-01-01', 6, 6),
147 (27, 'The Grapes of Wrath', '1939-04-14', 7, 1),
148 (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 Night''s Dream', '1596-01-01', 10, 7),
(31, 'The Tricky Book', '1888-01-01', 10, 7);
SELECT * FROM Material;
```

Data Output | Messages | Notifications

	material_id [PK] integer	title character varying (250)	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	The 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 Night's Dream	1596-01-01	10	7
31	31	The Tricky Book	1888-01-01	10	7

Total rows: 31 of 31 | Query complete 00:00:00.116 | Ln 153, Col 1

Member:

```
INSERT INTO Member (Member_ID, 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;
```

Data Output Messages Notifications

	member_id [PK] integer	name character varying (400)	contact_info character varying (200)	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	5	Emily Miller	emily.miller@email.com	2019-02-12
6	6	Frank Davis	frank.davis@email.com	2019-05-25
7	7	Grace Wilson	grace.wilson@email.com	2019-08-15
8	8	Harry Garcia	harry.garcia@email.com	2019-11-27
9	9	Isla Thomas	isla.thomas@email.com	2020-03-04
10	10	Jack Martinez	jack.martinez@email.com	2020-07-01
11	11	Kate Anderson	kate.anderson@email.com	2020-09-30
12	12	Luke Jackson	luke.jackson@email.com	2021-01-18
13	13	Mia White	mia.white@email.com	2021-04-27
14	14	Noah Harris	noah.harris@email.com	2021-07-13
15	15	Olivia Clark	olivia.clark@email.com	2021-10-05
16	16	Peter Lewis	peter.lewis@email.com	2021-12-01
17	17	Quinn Hall	quinn.hall@email.com	2022-02-28
18	18	Rachel Young	rachel.young@email.com	2022-06-17
19	19	Sam Walker	sam.walker@email.com	2022-09-25
20	20	Tiffany Allen	tiffany.allen@email.com	2022-12-10

Total rows: 20 of 20 Query complete 00:00:00.058 Ln 175, Col 30

Staff:

```
179 INSERT INTO Staff (Staff_ID, Name, Contact_Info, Job_Title, Hire_Date)
180 VALUES(1, 'Amy Green', 'amy.green@email.com', 'Librarian', '2017-06-01'),
181 (2, 'Brian Taylor', 'brian.taylor@email.com', 'Library Assistant', '2018-11-15'),
182 (3, 'Christine King', 'chris.king@email.com', 'Library Assistant', '2019-05-20'),
183 (4, 'Daniel Wright', 'dan.wright@email.com', 'Library Technician', '2020-02-01');
184 SELECT * FROM Staff;
185
186
```

Data Output Messages Notifications

	staff_id [PK] integer	name character varying (400)	contact_info character varying (200)	job_title character varying (255)	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

Total rows: 4 of 4 Query complete 00:00:00.060 Ln 186, Col 1

Borrow:

Query | Query History

```
186
187 INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)
188 VALUES
189 (1, 1, 1, 1, '2018-09-12', '2018-10-03', '2018-09-30'),
190 (2, 2, 2, 1, '2018-10-15', '2018-11-05', '2018-10-29'),
191 (3, 3, 3, 1, '2018-12-20', '2019-01-10', '2019-01-08'),
192 (4, 4, 4, 1, '2019-03-11', '2019-04-01', '2019-03-27'),
193 (5, 5, 5, 1, '2019-04-20', '2019-05-11', '2019-05-05'),
194 (6, 6, 6, 1, '2019-07-05', '2019-07-26', '2019-07-21'),
195 (7, 7, 7, 1, '2019-09-10', '2019-10-01', '2019-09-25'),
196 (8, 8, 8, 1, '2019-11-08', '2019-11-29', '2019-11-20'),
197 (9, 9, 9, 1, '2020-01-15', '2020-02-05', '2020-02-03'),
198 (10, 10, 10, 1, '2020-03-12', '2020-04-02', '2020-03-28'),
199 (11, 1, 11, 2, '2020-05-14', '2020-06-04', '2020-05-28'),
200 (12, 2, 12, 2, '2020-07-21', '2020-08-11', '2020-08-02'),
201 (13, 3, 13, 2, '2020-09-25', '2020-10-16', '2020-10-15'),
202 (14, 4, 1, 2, '2020-11-08', '2020-11-29', '2020-11-24'),
203 (15, 5, 2, 2, '2021-01-03', '2021-01-24', '2021-01-19'),
204 (16, 6, 3, 2, '2021-02-18', '2021-03-11', '2021-03-12'),
205 (17, 17, 4, 2, '2021-04-27', '2021-05-18', '2021-05-20'),
206 (18, 18, 5, 2, '2021-06-13', '2021-07-04', '2021-06-28'),
207 (19, 19, 6, 2, '2021-08-15', '2021-09-05', '2021-09-03'),
208 (20, 20, 7, 2, '2021-10-21', '2021-11-11', '2021-11-05'),
209 (21, 21, 1, 3, '2021-11-29', '2021-12-20', NULL),
210 (22, 22, 2, 3, '2022-01-10', '2022-01-31', '2022-01-25'),
211 (23, 23, 3, 3, '2022-02-07', '2022-02-28', '2022-02-23'),
212 (24, 24, 4, 3, '2022-03-11', '2022-04-01', '2022-03-28'),
213 (25, 25, 5, 3, '2022-04-28', '2022-05-19', '2022-05-18'),
214 (26, 26, 6, 3, '2022-06-22', '2022-07-13', '2022-07-08'),
```

```
215 (27, 27, 7, 3, '2022-08-04', '2022-08-25', '2022-08-23'),
216 (28, 28, 8, 3, '2022-09-13', '2022-10-04', '2022-09-28'),
217 (29, 29, 9, 3, '2022-10-16', '2022-11-06', '2022-11-05'),
218 (30, 30, 8, 3, '2022-11-21', '2022-12-12', '2022-12-05'),
219 (31, 1, 9, 4, '2022-12-28', '2023-01-18', NULL),
220 (32, 2, 1, 4, '2023-01-23', '2023-02-13', NULL),
221 (33, 3, 10, 4, '2023-02-02', '2023-02-23', '2023-02-17'),
222 (34, 4, 11, 4, '2023-03-01', '2023-03-22', NULL),
223 (35, 5, 12, 4, '2023-03-10', '2023-03-31', NULL),
224 (36, 6, 13, 4, '2023-03-15', '2023-04-05', NULL),
225 (37, 7, 17, 4, '2023-03-25', '2023-04-15', NULL),
226 (38, 8, 8, 4, '2023-03-30', '2023-04-20', NULL),
227 (39, 9, 9, 4, '2023-03-26', '2023-04-16', NULL),
228 (40, 10, 20, 4, '2023-03-28', '2023-04-18', NULL);
229 SELECT * FROM Borrow;
```

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

Total rows: 40 of 40 Query complete 00:00:00.065 Ln 223, Col 50

Author:

```
INSERT INTO Author (Author_ID, 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 Brontë', '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;
```

Data Output | Messages | Notifications

	author_id [PK] integer	name character varying (400)	birth_date date	nationality character varying (100)
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 Brontë	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

Total rows: 20 of 20 | Query complete 00:00:00.067 | Ln 253, Col 22

Authorship:

Query | Query History

```
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;
```

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

Total rows: 33 of 33 | Query complete 00:00:00.060 | Ln 278, Col 13

Querying and Manipulation:

Queries/Updates

1. Which materials are currently available in the library?

Query | Query History

```
SELECT * FROM Material
WHERE Material_ID NOT IN (SELECT Material_ID FROM Borrow WHERE Return_Date IS NULL);
```

Data Output | Messages | Notifications

	material_id [PK] integer	title character varying (250)	publication_date date	catalog_id integer	genre_id integer
1	3	The Da Vinci Code	2003-04-01	3	2
2	11	1984	1949-06-08	1	5
3	12	Animal Farm	1945-08-17	2	5
4	13	The Haunting of Hill House	1959-10-17	3	4
5	14	Brave New World	1932-08-01	4	5
6	15	The Chronicles of Narnia: The Lion, the Witch and the Wardro...	1950-10-16	5	3
7	16	The Adventures of Huckleberry Finn	1884-12-10	6	1
8	17	The Catch-22	1961-10-11	7	1
9	18	The Picture of Dorian Gray	1890-07-01	8	1
10	19	The Call of Cthulhu	1928-02-01	9	4
11	20	Harry Potter and the Philosopher's Stone	1997-06-26	10	3
12	22	A Tale of Two Cities	1859-04-30	7	1
13	23	The Iliad	1750-01-01	8	6
14	24	The Odyssey	1725-01-01	9	6
15	25	The Brothers Karamazov	1880-01-01	10	1
16	26	The Divine Comedy	1320-01-01	6	6
17	27	The Grapes of Wrath	1939-04-14	7	1
18	28	The Old Man and the Sea	1952-09-01	8	1
19	29	The Count of Monte Cristo	1844-01-01	9	1
20	30	A Midsummer Night's Dream	1596-01-01	10	7
21	31	The Tricky Book	1888-01-01	10	7

Total rows: 21 of 21 | Query complete 00:00:00.080 | Ln 3, Col 1

This query will select all materials whose Material_ID does not appear in the Borrow table with a null Return_Date, indicating that the material has not been returned yet, so remaining other materials are currently available materials in the library.

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

```
Q2
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';
```

Data Output | Messages | Notifications

	title character varying (250)	borrow_date date	due_date date
1	The Catcher in the Rye	2022-12-28	2023-01-18
2	To Kill a Mockingbird	2023-01-23	2023-02-13
3	The Hobbit	2023-03-01	2023-03-22
4	The Shining	2023-03-10	2023-03-31
5	Frankenstein	2021-11-29	2021-12-20

This query will return a list of all materials that are currently borrowed, have a due date prior to 04/01/2023, and have not yet been returned i.e. (overdue), along with their respective borrow dates and due dates.

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.

```
Q3
SELECT m.title, COUNT(*) 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;
```

Data Output | Messages | Notifications

	title character varying (250)	available_count bigint
1	The Catcher in the Rye	3
2	Pride and Prejudice	3
3	The Da Vinci Code	3
4	The Hobbit	3
5	To Kill a Mockingbird	3
6	The Shining	3
7	The Great Gatsby	2
8	Moby Dick	2
9	Crime and Punishment	2
10	The Hitchhiker's Guide to the Galaxy	2

Total rows: 10 of 10 | Query complete 00:00:00.054 | Ln 18, Col 1

This query joins the material and borrow tables on the material_id field, filters out materials that have been returned by checking for a null return_date, groups the results by the title attritibe, and then orders them in descending order of available_count. Finally, we limit the results to the top 10 rows as the top 10 most borrowed materials in the library.

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

```
22  Q4
23  Select COUNT(*) AS Books_Written
24  FROM material m, authorship a, author b
25  where (b.author_id= a.author_id) AND (m.material_Id=a.material_Id) AND (b.name='Lucas Piki');
26
```

Data Output | Messages | Notifications

	books_written bigint
1	1

This query will return the number of books written by Lucas Piki.

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

```
28  Q5
29  SELECT COUNT(material_id) As Num_of_Books_Writtenby_2_or_more_Authors
30  FROM (
31    SELECT material_id, COUNT(DISTINCT author_id) AS Books
32    FROM authorship
33    GROUP BY material_id
34    HAVING COUNT(DISTINCT Authorship.Author_ID) >= 2
35  ) AS Books;
```

Data Output | Messages | Notifications

	num_of_books_writtenby_2_or_more_authors bigint
1	3

This query retrieves number of books written by two or more authors.

Also, we can retrieve the names of books that were written by two or more authors as below query:

```
43
44  SELECT m.title, COUNT(DISTINCT a.author_id) AS Authors
45  FROM material m
46  JOIN authorship a ON m.material_id = a.material_id
47  GROUP BY m.material_id, m.title
48  HAVING COUNT(DISTINCT a.author_id) >= 2;
```

Data Output | Messages | Notifications

	title character varying (250)	authors bigint
1	The Old Man and the Sea	2
2	The Count of Monte Cristo	2
3	A Midsummer Night's Dream	2

This query will return the title of each book that was written by two or more authors, along with the number of unique authors for that book.

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

```
51  Q6
52  SELECT Genre.Name, COUNT(*) AS popular_genres
53  FROM Genre
54  INNER JOIN Material ON Material.Genre_ID = Genre.Genre_ID
55  GROUP BY Genre.Name
56  ORDER BY popular_genres DESC;
57
```

Data Output | Messages | Notifications

	name character varying (400)	popular_genres bigint
1	General Fiction	14
2	Science Fiction & Fantasy	4
3	Horror & Suspense	4
4	Dystopian & Apocalyptic	3
5	Classics	3
6	Historical Fiction	2
7	Mystery & Thriller	1

This query joins the Material & Genre tables on the Genre_ID column and groups the results by the genre name. It then counts the number of rows in each group using the COUNT (*) function and sorts the results in descending order of count by DESC.
Therefore, the most popular genre in the library is General Fiction with 14 books.

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

```
59  Q7
60  SELECT COUNT(*) AS Borrowed_Materials
61  FROM Borrow
62  WHERE borrow_date BETWEEN '2020-09-01' AND '2020-10-31';
```

Data Output | Messages | Notifications

	borrowed_materials bigint
1	1

This query uses the COUNT () function to count the number of rows returned by the query as Borrowed_Materials. And the WHERE clause filters the results to include only rows where the borrow_date is between 2020-09-01 to 2020-10-31.

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

```
64  Q8
65  UPDATE borrow
66  SET return_date = '2023-04-01'
67  WHERE material_id = (SELECT material_id FROM material WHERE title = 'Harry Potter and the Philosophers Stone');
68  select * from borrow;
```

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
31	32	2	1	4	2023-01-23	2023-02-13	[null]
32	33	3	10	4	2023-02-02	2023-02-23	2023-02-17
33	34	4	11	4	2023-03-01	2023-03-22	[null]
34	35	5	12	4	2023-03-10	2023-03-31	[null]
35	36	6	13	4	2023-03-15	2023-04-05	[null]
36	37	7	17	4	2023-03-25	2023-04-15	[null]
37	38	8	8	4	2023-03-30	2023-04-20	[null]
38	39	9	9	4	2023-03-26	2023-04-16	[null]
39	40	10	20	4	2023-03-28	2023-04-18	[null]
40	20	20	7	2	2021-10-21	2021-11-11	2023-04-01

Total rows: 40 of 40 | Query complete 00:00:00.082 | Ln 67, Col 112

This query updates the return_date of the corresponding record in the borrow table. The subquery in the WHERE clause of the update statement retrieves the title based on the material_id.

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

```
67  Q9
68  DELETE FROM borrow
69  WHERE member_id IN (SELECT member_id FROM member WHERE name = 'Emily Miller');
70  DELETE FROM member WHERE name = 'Emily Miller';
71  Select * from borrow;
72
```

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

Total rows: 37 of 37 | Query complete 00:00:00.280 | Ln 72, Col 1

The first query deletes all the records from the borrow table that have a matching "member_id" with the "member_id" of Emily Miller in the member table. The second query deletes the record of Emily Miller from the member table.

10. How do you add the following material to the database?

```
73  Q10
74  INSERT INTO material (material_ID,Title, Publication_Date, catalog_ID, genre_ID)
75  VALUES (32,'New book', '2020-08-01',
76          (SELECT catalog_ID FROM Catalog WHERE Name = 'E-Books'),
77          (SELECT genre_ID FROM Genre WHERE Name = 'Mystery & Thriller'));
78  INSERT INTO author (author_id, name, birth_date, nationality) VALUES (21,'Lucas Pipi', null, null);
79
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 52 msec.
```

This query inserted the given material into the tables.

```
82  --Here I try to retrive the material which was added earlier
83
84  SELECT m.title, m.Publication_Date, c.name, g.name
85  FROM material m
86  INNER JOIN catalog c ON c.catalog_ID = m.catalog_ID
87  INNER JOIN Genre g ON g.genre_ID = m.genre_ID
88  where title='New book';
```

Data Output | Messages | Notifications

	title character varying (250)	publication_date date	name character varying (180)	name character varying (400)
1	New book	2020-08-01	E-Books	Mystery & Thriller

So, I tried to retrieve the given material from the database which was added earlier.

Design existing Database:

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

- To extend an existing database system to integrate the two functions, we need to add some tables to the database and modify existing tables.
- We can create a new table called "Overdue_Materials" with the following columns:

  MemberID (foreign key referencing Members table)
  MaterialID (foreign key referencing Materials table)
  ReturnDate
  DueDate
  After creating table, we can create a stored procedure that checks this table daily and sends alerts to staff if any material is overdue.

I have given the query in text box as below:

```
CREATE PROCEDURE Check_Overdue_Materials as

BEGIN

  DECLARE today DATE = GETDATE()

  SELECT mem.MemberName, mat.MaterialName, du.DueDate
  FROM Overdue_Materials k
  INNER JOIN Members mem ON k.MemberID = mem.MemberID
  INNER JOIN Materials ma ON k.MaterialID = mat.MaterialID
  WHERE k.ReturnDate IS NULL AND k.DueDate < today
END
```

This query of stored procedure returns the names of the members with overdue material, the names of the overdue materials, and the due date of the materials.

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.

- We can create a trigger to deactivate & reactivate membership with following steps with queries.

Step 1:
Create a new table to keep track of the number of overdue occurrences for each member: Sql.

```
CREATE TABLE member_over_due (
  member_id INT PRIMARY KEY,
  over_due_count INT DEFAULT 0
);
```

Step 2:
After creating table, we now create a trigger that updates the over_due_count in the member_over_due table whenever a member checks out a material.

```
CREATE TRIGGER update_over_due_count
AFTER INSERT ON checkout FOR EACH ROW
BEGIN
  ---- Increase over_due_count by 1 if the material is overdue
  IF N.due_date < CDATE() THEN UPDATE member_over due
    SET over_due_count = over_due_count + 1
    WHERE member_id = N.member_id;
  END IF;
END;
```

.

Step 3:
After creating the update_over_due_count trigger, now we create a trigger that deactivates a membership if the member has overdue occurrences >= three times (3).

```
CREATE TRIGGER deactivate_membership
AFTER UPDATE ON member_over_due FOR EACH ROW
BEGIN
  -- Deactivate membership if overdue occurrences >= 3
  IF N.overdue_count >= 3 THEN
    UPDATE member SET active = false
    WHERE member_id = N.member_id;
  END IF;
END;
```

Step 4:
After creating both the triggers, now we finally create a trigger that reactivates a membership once the member pays the overdue fee.

```
CREATE TRIGGER reactivate_membership
AFTER UPDATE ON member FOR EACH ROW
BEGIN
  --Reactivate membership if the member pays the
overdue fee
  IF N.active = true THEN
    UPDATE member_overdue SET overdue_count = 0
    WHERE member_id = N.member_id;
  END IF;
END;
```

- By all these triggers which we are used in the database it will automatically deactivate the membership based on the member's overdue occurrence (>= three times). And reactivate the membership once the member pays the overdue fee.