MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

OVERVIEW:

The Library Management System project is a complete software solution aimed to improve the efficiency of library operations. It enables the automation of many processes within a library or literary resource center. This system attempts to simplify the management of various resources such as books, magazines, e-books, and other products. This not only enhances the overall experience for both library users and staff, but it also strives to speed resource management operations.

SCOPE:

This project involves designing and implementing a database management system for a public library. The system will enable efficient management of library materials, member information, and borrowing transactions and provide reporting and analytics capabilities. Key entities include Material, Catalog, Genre, Borrow, Author, Authorship, Member, and Staff, with defined attributes and relationships to support library operations.

- **Cataloging of Materials:**
  - Manage diverse materials: books, magazines, e-books, and audiobooks.
  - Include details: title, publication date, genre, and catalog location.
- **Member Management:**
  - Maintain member records: names, contact info, and membership start date.
  - Facilitate registration and account management.
- **Borrowing and Returns:**
  - Enable material borrowing with issue and due date tracking.
  - Manage returns and update material availability.
- **Author and Authorship Tracking:**
  - Record author details: names, birth dates, and nationalities.
  - Establish authorship relationships with respective works.
- **Staff Management:**
  - Manage staff information: job titles, contact info, and hire dates.
  - Assign roles for operational efficiency.
- **Search and Query Capability:**
  - Allow users to search materials using criteria (title, author, genre).
  - Facilitate advanced queries for administrative reports.

ENTITIES:

- Material:
  Represents items available in the library such as books, e-books, etc.

  Attributes:
  - Material_ID (unique identifier)
  - Title
  - Publication_Date
  - Catalog_ID (a reference to the catalog entry for the material)
  - Genre_ID (a reference to the genre of the material)

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

- Catalog:
  Records the availability and location of library materials.

  Attributes:
  - Catalog_ID (unique identifier)
  - Name
  - Location

- Genre:
  Categorizes materials into different genres like Fiction, Non-fiction, etc.

  Attributes:
  - Genre_ID (unique identifier)
  - Name
  - Description

- Borrow:
  Details the borrowing activity of members, including dates and staff involvement.

  Attributes:
  - Borrow_ID (unique identifier)
  - Material_ID (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
  - Due_Date
  - Return_Date

- Author:
  Contains information about the authors of the materials.

  Attributes:
  - Author_ID (unique identifier)
  - Name
  - Birth_Date
  - Nationality

- Authorship:
  Associate authors with the materials they have created.

  Attributes:
  - Authorship_ID (unique identifier)
  - Author_ID (reference to the author)
  - Material_ID (a reference to the material authored)

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

- Member:
  Represents the library members who borrow materials.

  Attributes:
  - Member_ID (unique identifier)
  - Name
  - Contact_Info
  - Join_Date

- Staff:
  Represents the employees of the library responsible for managing various tasks.

  Attributes:
  - Staff_ID (unique identifier)
  - Name
  - Contact_Info
  - Job_Title
  - Hire_Date

RELATIONSHIP:

- **MATERIAL BELONGS TO CATALOG**:
  It shows Many to One relationship. Each catalog has many material entry, but each material is linked to one catalog entry.

  Constraints:
  - Material has total participation i.e. every material will belong to some catalog.
  - Catalog has a partial participation i.e. every catalog does not require to have a material.

- **MATERIAL GROUP BY GENRE:**
  It shows Many to One relationship. Each genre is linked to many material entry, but each material is linked to one genre entry.

  Constraints:
  - Material has total participation i.e. every material will belong to some genre.
  - Catalog has a partial participation i.e. every genre does not require to have a material.

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

- **MATERIAL AUTHORSHIP OF AUTHORSHIP:**
  It displays Many to One relationship. Where each authorship is linked to many material entry, but each material is linked to one authorship entry.

  Constraints:
  - Material and Authorship both have total participation i.e. every material will belong to some authorship, Vice versa, every authorship will belong to some material.

- **AUTHORSHIP AUTHORED BY AUTHOR:**
  It shows Many to One relationship. Each author is linked to many authorship entry, but each authorship is linked to one author entry.

  Constraints:
  - Authorship has total participation i.e. every authorship will belong to some author.
  - Author has a partial participation i.e. every author does not require to have a authorship.

- **BORROW CHECKED OUT MATERIAL:**
  It shows one to many relationship. Every material is linked to one borrow entry, whereas each borrow is linked to many material entry.

  Constraints:
  - Borrow has total participation i.e. every borrow will belong to some material.
  - Material has a partial participation i.e. every material does not require to have a borrow.

- **BORROW PROCESSED BY STAFF:**
  It shows many to one relationship. Every borrow is linked to one staff entry, whereas each staff is linked to many borrow entry.

  Constraints:
  - Borrow has total participation i.e. every borrow will belong to some staff.
  - Staff has a partial participation i.e. every staff does not require to have a borrow.

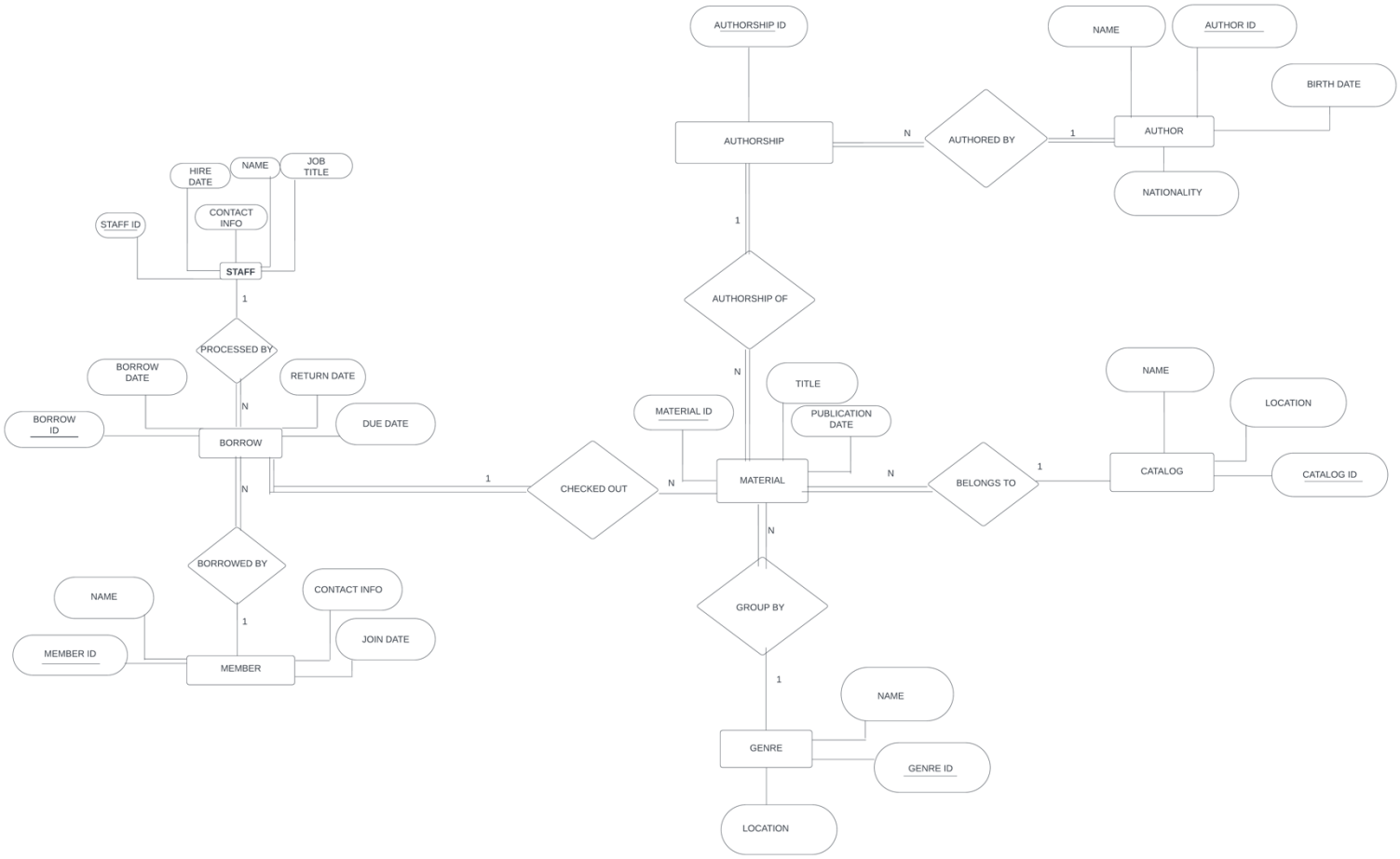
- **BORROW BORROWED BY MEMBER:**
  It shows many to one relationship. Every borrow is linked to one member entry, whereas each borrow is linked to many member entry.

  Constraints:
  - Borrow has total participation i.e. every borrow will belong to some member.
  - Member has a partial participation i.e. every member does not require to have a borrow.

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

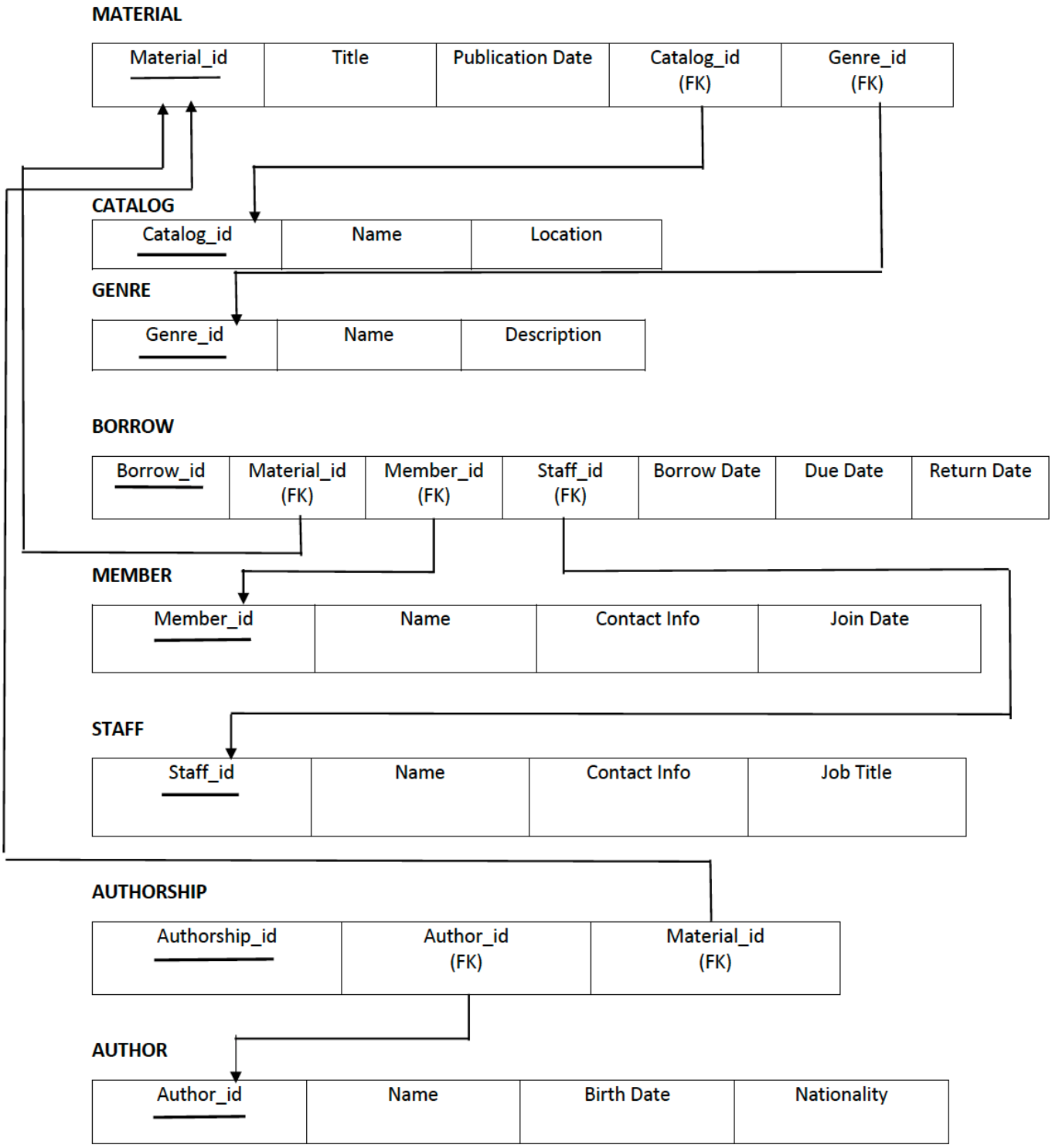
ENTITY – RELATIONSHIP DIAGRAM:



MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

RELATIONAL SCHEMA:



MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

DATABASE IMPLEMENTATION:

PostgreSQL was chosen as our major database management system (DBMS) for the implementation of our library management system. PostgreSQL, a powerful open-source object-relational database, combines robust features, high dependability, and good performance, making it an excellent solution for our project's requirements.

SCHEMA CREATION:

Material:

```
-- Create sequence for material_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE material_sequence
  start 1
  increment 1;

-- Create Material table
CREATE TABLE Material (
  Material_ID INTEGER PRIMARY KEY,
  Title TEXT,
  Publication_Date DATE,
  Catalog_ID INTEGER,
  Genre_ID INTEGER,
  FOREIGN KEY (Catalog_ID) REFERENCES Catalog(Catalog_ID),
  FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);
```

Catalog:

```
-- Create sequence for catalog_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE catalog_sequence
  start 1
  increment 1;

-- Create Catalog table
CREATE TABLE Catalog (
  Catalog_ID INTEGER PRIMARY KEY,
  Name TEXT,
  Location TEXT
);
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

Genre:

```
-- Create sequence for genre_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE genre_sequence
  start 1
  increment 1;

-- Create Genre table
CREATE TABLE Genre (
  Genre_ID INTEGER PRIMARY KEY,
  Name TEXT,
  Description TEXT
);
```

Borrow:

```
-- Create sequence for borrow_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE borrow_sequence
  start 1
  increment 1;

-- Create Borrow table
CREATE TABLE Borrow (
  Borrow_ID INTEGER PRIMARY KEY,
  Material_ID INTEGER,
  Member_ID INTEGER,
  Staff_ID INTEGER,
  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)
);
```

Author:

```
-- Create sequence for author_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE author_sequence
  start 1
  increment 1;
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

```
-- Create Author table
CREATE TABLE Author (
  Author_ID INTEGER PRIMARY KEY,
  Name TEXT,
  Birth_Date DATE,
  Nationality TEXT
);
```

Authorship:

```
-- Create sequence for authorship_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE authorship_sequence
  start 1
  increment 1;

-- Create Authorship table
CREATE TABLE Authorship (
  Authorship_ID INTEGER PRIMARY KEY,
  Author_ID INTEGER,
  Material_ID INTEGER,
  FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID),
  FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID)
);
```

Member:

```
-- Create sequence for member_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE member_sequence
  start 1
  increment 1;

-- Create Member table
CREATE TABLE Member (
  Member_ID INTEGER PRIMARY KEY,
  Name TEXT,
  Contact_Info TEXT,
  Join_Date DATE
);
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

Staff:

```
-- Create sequence for staff_id
-- The sequence generates unique ID for each entry
CREATE SEQUENCE staff_sequence
  start 1
  increment 1;

-- Create Staff table
CREATE TABLE Staff (
  Staff_ID INTEGER PRIMARY KEY,
  Name TEXT,
  Contact_Info TEXT,
  Job_Title TEXT,
  Hire_Date DATE
);
```

DATABASE POPULATION:

Material:

```
-- Load data into Material table
COPY material(material_id, title, publication_date, catalog_id,
genre_id)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Material.csv'
DELIMITER ','
CSV HEADER;

-- Set material_sequence value to the last id after import
SELECT setval('material_sequence', (SELECT MAX(material_id) FROM
Material))
```

Catalog:

```
-- Load data into Catalog table
COPY catalog(catalog_id, name, location)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Catalog.csv'
DELIMITER ','
CSV HEADER;

-- Set catalog_sequence value to the last id after import
SELECT setval('catalog_sequence', (SELECT MAX(catalog_id) FROM
"catalog"))
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

Genre:

```
-- Load data into Genre table
COPY genre(genre_id, name, description)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Genre.csv'
DELIMITER ','
CSV HEADER;

-- Set genre_sequence value to the last id after import
SELECT setval('genre_sequence', (SELECT MAX(genre_id) FROM
genre))
```

Borrow:

```
-- Load data into Borrow table
COPY borrow(borrow_id, material_id, member_id, staff_id,
borrow_date, due_date, return_date)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Borrow.csv'
DELIMITER ','
CSV HEADER;

-- Set borrow_sequence value to the last id after import
SELECT setval('borrow_sequence', (SELECT MAX(borrow_id) FROM
borrow))
```

Author:

```
-- Load data into Author table
COPY author(author_id, name, birth_date, nationality)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Author.csv'
DELIMITER ','
CSV HEADER;

-- Set author_sequence value to the last id after import
SELECT setval('author_sequence', (SELECT MAX(author_id) FROM
author))
```

Authorship:

```
-- Load data into Authorship table
COPY authorship(authorship_id, author_id, material_id)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Authorship.csv'
DELIMITER ','
CSV HEADER;
```

CS 504 - PROJECT

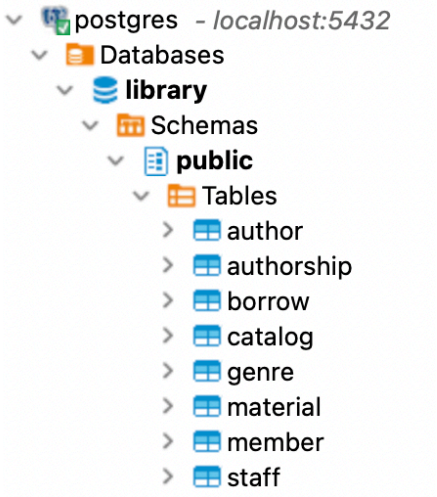
```
-- Set authorship_sequence value to the last id after import
SELECT setval('authorship_sequence', (SELECT MAX(Authorship_ID)
FROM Authorship))
```

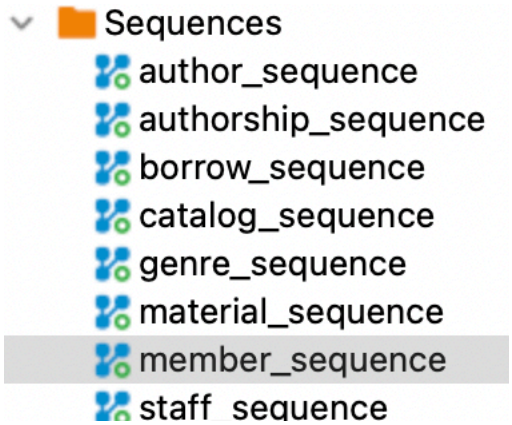
Member:

```
-- Load data into Member table
COPY member(member_id, name, contact_info, join_date)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Member.csv'
DELIMITER ','
CSV HEADER;

-- Set member_sequence value to the last id after import
SELECT setval('member_sequence', (SELECT MAX(member_id) FROM
"member"))
```

Staff:

```
-- Load data into Staff table
COPY staff(staff_id, name, contact_info, job_title, hire_date)
FROM '/Users/mukesh/Desktop/GMU/CS 504/Project/Staff.csv'
DELIMITER ','
CSV HEADER;

-- Set staff_sequence value to the last id after import
SELECT setval('staff_sequence', (SELECT MAX(staff_id) FROM
staff))
```

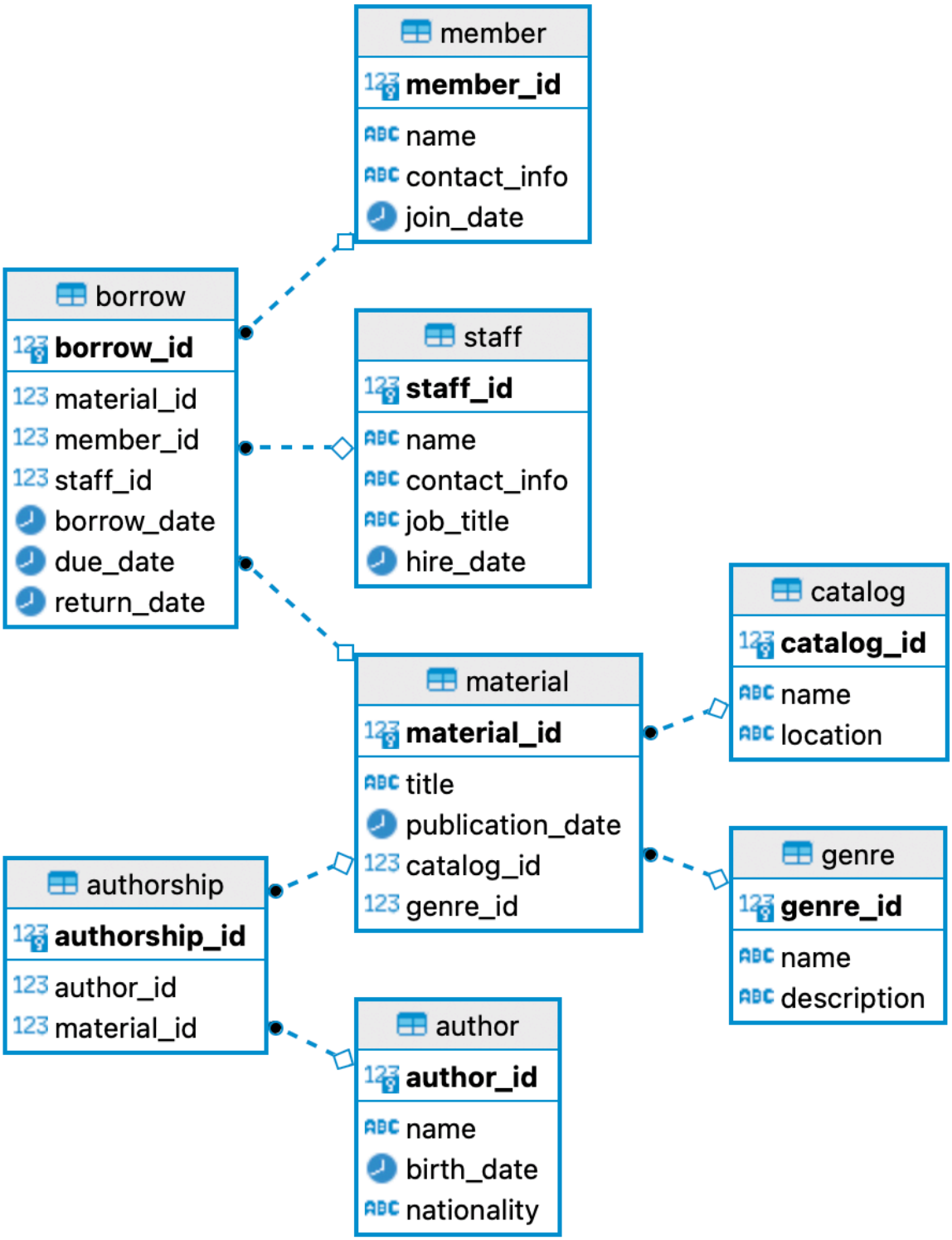




MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

SCHEMA DIAGRAM:



MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

QUERYING AND MANIPULATION:

Searching:

```
--Searching
SELECT * FROM Material WHERE Title LIKE '%Rye%';
```

SELECT * FROM Material WHERE Title | Enter a SQL expression to filter results (use Ctrl+Space)

	material_id	title	publication_date	catalog_id	genre_id
1	1	The Catcher in the Rye	1951-07-16	1	1

Inserting:

```
--Inserting
INSERT INTO Member (member_id, name, contact_info, join_date)
VALUES (nextval('member_sequence'), 'Mukesh Rajmohan', 'mrajmoha@gmu.edu.in', CURRENT_DATE);

select * from member where name = 'Mukesh Rajmohan';
```

select * from member where name = 'M | Enter a SQL expression to filter results (use Ctrl+Space)

	member_id	name	contact_info	join_date
1	21	Mukesh Rajmohan	mrajmoha@gmu.edu.in	2023-11-14

Updating:

```
--Updating
UPDATE Member SET Contact_Info = 'mukeshrmohan@gmail.com' WHERE Member_ID = 21;

select * from member where name = 'Mukesh Rajmohan';
```

select * from member where name = 'M | Enter a SQL expression to filter results (use Ctrl+Space)

	member_id	name	contact_info	join_date
1	21	Mukesh Rajmohan	mukeshrmohan@gmail.com	2023-11-14

Deleting:

```
--Deleting
DELETE FROM member WHERE member_id = 21;

select * from member where name = 'Mukesh Rajmohan';
```

select * from member where name = 'M | Enter a SQL expression to filter results (use C

	member_id	name	contact_info	join_date

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

Joins:

```
--Joins
SELECT m.Title, a.Name as author_name
FROM Material m JOIN Authorship au ON m.Material_ID = au.Material_ID
JOIN Author a ON au.Author_ID = a.Author_ID;
```

material(+) 1

SELECT m.Title, a.Name as author_nar | Enter a SQL expression to filter results (use Ctrl+Spa

	title	author_name
1	The Catcher in the Rye	Jane Austen
2	To Kill a Mockingbird	Ernest Hemingway
3	The Da Vinci Code	George Orwell
4	The Hobbit	Scott Fitzgerald
5	The Shining	J.K. Rowling
6	Pride and Prejudice	Mark Twain
7	The Great Gatsby	Leo Tolstoy
8	Moby Dick	Virginia Woolf
9	Crime and Punishment	Gabriel Márquez
10	The Hitchhiker's Guide to the Galaxy	Charles Dickens

Aggregation:

```
-- Aggregation
SELECT g.Name, COUNT(m.Genre_ID) AS NumberOfBooks
FROM Material m JOIN Genre g
ON m.Genre_ID = g.Genre_ID
GROUP BY g.Name;
```

genre 1

SELECT g.Name, COUNT(m.Genre_ID) | Enter a SQL expression t

	name	numberofbooks
1	General Fiction	14
2	Classics	3
3	Horror & Suspense	4
4	Dystopian & Apocalyptic	3
5	Historical Fiction	2
6	Mystery & Thriller	1
7	Science Fiction & Fantasy	4

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

Subqueries:

```
-- Subqueries
SELECT m.Name
FROM Member m
WHERE m.Member_ID IN
(SELECT Member_ID
FROM Borrow
GROUP BY Member_ID HAVING COUNT(Material_ID) > 2);
```

member 1

SELECT m.Name FROM Member m WH | Enter a SQL expression to fi

	name
1	Alice Johnson
2	Bob Smith
3	Carol Brown
4	David Williams
5	Frank Davis
6	Grace Wilson
7	Harry Garcia
8	Isla Thomas

QUERIES / UPDATES:

1. **Which materials are currently available in the library? If a material is borrowed and not returned, it's not considered available.**

   Query:

```
SELECT material_id,title
FROM Material
WHERE Material_ID NOT IN (
      SELECT Material_ID
      FROM Borrow
      WHERE Return_Date IS NULL
);
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

```sql
-- Query 1
SELECT material_id,title
FROM Material
WHERE Material_ID NOT IN (
    SELECT Material_ID
    FROM Borrow
    WHERE Return_Date IS NULL
  );
```

material 1

SELECT material_id,title FROM Materi | Enter a SQL expression to filter results (use Ctrl+S

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

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

Query:

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

CS 504 - PROJECT

```
-- Query 2
SELECT m.material_id,m.Title,b.Borrow_Date,b.Due_Date
FROM Borrow b
INNER JOIN Material m
ON b.Material_ID = m.Material_ID
WHERE b.Return_Date IS NULL AND b.Due_Date < '2023-04-01'
order by due_date;
```

material(+) 1

SELECT m.material_id,m.Title,b.Borrov | Enter a SQL expression to filter results (use Ctrl+Space)

	material_id	title	borrow_date	due_date
1	20	Harry Potter and the Philosopher's Stone	2021-10-21	2021-11-11
2	21	Frankenstein	2021-11-29	2021-12-20
3	1	The Catcher in the Rye	2022-12-28	2023-01-18
4	2	To Kill a Mockingbird	2023-01-23	2023-02-13
5	4	The Hobbit	2023-03-01	2023-03-22
6	5	The Shining	2023-03-10	2023-03-31

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.**

Query:

```
SELECT m.material_id,m.Title, COUNT(b.Material_ID) AS total_borrows
FROM Material m
INNER JOIN Borrow b
ON b.Material_ID = m.Material_ID
GROUP BY m.material_id,m.Title
ORDER BY total_borrows DESC
LIMIT 10;
```

```
-- Query 3
SELECT m.material_id,m.Title, COUNT(b.Material_ID) AS total_borrows
FROM Material m
INNER JOIN Borrow b
ON b.Material_ID = m.Material_ID
GROUP BY m.material_id,m.Title
ORDER BY total_borrows DESC
LIMIT 10;
```

material 1

SELECT m.material_id,m.Title, COUNT | Enter a SQL expression to filter results (use Ctrl-

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

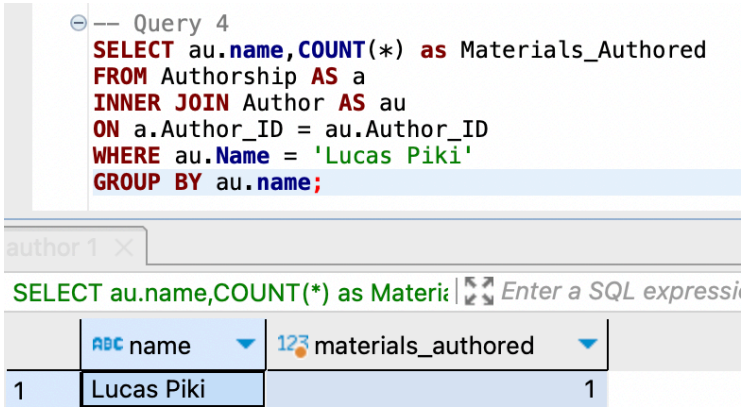
MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

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

Query:

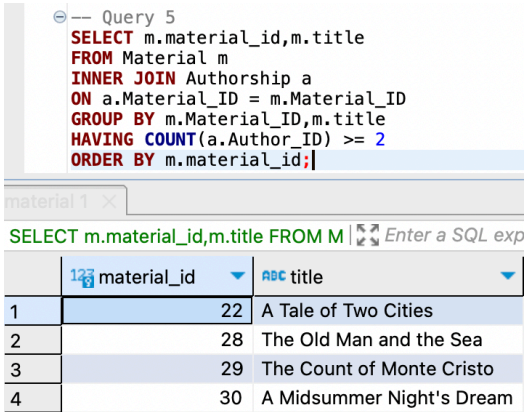
```
SELECT au.name,COUNT(*) as Materials_Authored
FROM Authorship AS a
INNER JOIN Author AS au
ON a.Author_ID = au.Author_ID
WHERE au.Name = 'Lucas Piki'
GROUP BY au.name;
```



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

Query:

```
SELECT m.material_id,m.title
FROM Material m
INNER JOIN Authorship a
ON a.Material_ID = m.Material_ID
GROUP BY m.Material_ID,m.title
HAVING COUNT(a.Author_ID) >= 2
ORDER BY m.material_id;
```



MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

6. **What are the most popular genres in the library ranked by the total number of borrowed times of each genre?**

Query:

```
SELECT g.Name,COUNT(b.Borrow_ID) AS total_borrows
FROM Genre g
INNER JOIN Material m ON g.Genre_ID = m.Genre_ID
INNER JOIN Borrow b ON m.Material_ID = b.Material_ID
GROUP BY g.Name
ORDER BY total_borrows DESC;
```

```
-- Query 6
SELECT g.Name,COUNT(b.Borrow_ID) AS total_borrows
FROM Genre g
INNER JOIN Material m ON g.Genre_ID = m.Genre_ID
INNER JOIN Borrow b ON m.Material_ID = b.Material_ID
GROUP BY g.Name
ORDER BY total_borrows DESC;
```

genre 1

SELECT g.Name,COUNT(b.Borrow_ID) | Enter a SQL expression to fi

	name	total_borrows
1	General Fiction	20
2	Science Fiction & Fantasy	6
3	Horror & Suspense	4
4	Classics	3
5	Mystery & Thriller	3
6	Historical Fiction	1

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

Query:

```
SELECT m.material_id,m.title
FROM Borrow b
INNER JOIN Material m
ON b.Material_ID = m.Material_ID
WHERE b.Borrow_Date BETWEEN '2020-09-01' AND '2020-10-31';
```

```
-- Query 7
SELECT m.material_id,m.title
FROM Borrow b
INNER JOIN Material m
ON b.Material_ID = m.Material_ID
WHERE b.Borrow_Date BETWEEN '2020-09-01' AND '2020-10-31';
```

material 1

SELECT m.material_id,m.title FROM B | Enter a SQL expression to filter re

	material_id	title
1	3	The Da Vinci Code

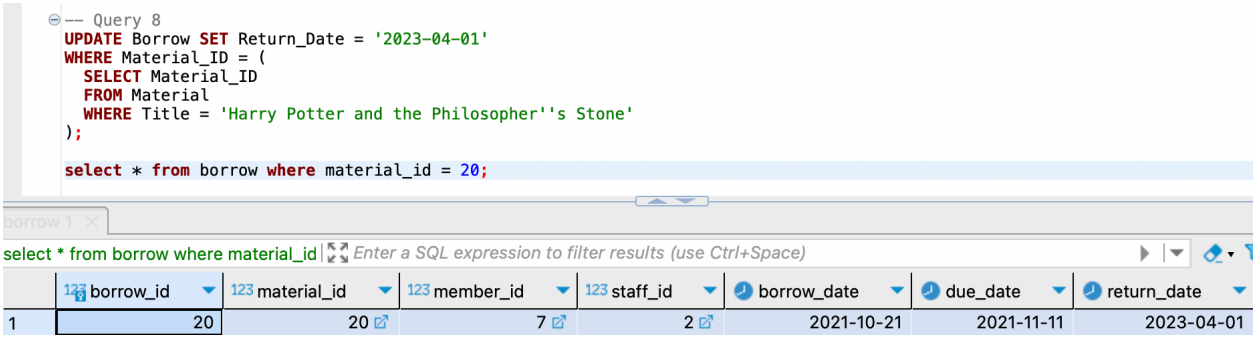
MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

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

Query:

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

select * from borrow where material_id = 20;
```



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

Query:

```
BEGIN;

-- Delete Emily's borrows
DELETE FROM Borrow
WHERE Member_ID = (
      SELECT Member_ID
      FROM Member
      WHERE Name = 'Emily Miller'
);

select * from borrow where member_id = 5;

-- Delete Emily from Member table
DELETE FROM Member
WHERE Name = 'Emily Miller';

select * from member where Name = 'Emily Miller';
COMMIT;
```

CS 504 - PROJECT

```
-- Query 9
BEGIN;

-- Delete Emily's borrows
DELETE FROM Borrow
WHERE Member_ID = (
  SELECT Member_ID
  FROM Member
  WHERE Name = 'Emily Miller'
);

select * from borrow where member_id = 5;

-- Delete Emily from Member table
DELETE FROM Member
WHERE Name = 'Emily Miller';

select * from member where Name = 'Emily Miller';

COMMIT;
```

member 1

select * from member where Name = 'E | Enter a SQL expression to filter results (use

	member_id	name	contact_info	join_date

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

Query:

```
BEGIN;

-- Insert new material
INSERT INTO Material (material_id, Title, Publication_Date, Catalog_ID,
Genre_ID)
VALUES (nextval('material_sequence'),'New Book', '2020-08-01',
  (SELECT Catalog_ID FROM Catalog WHERE Name = 'E-Books'),
  (SELECT Genre_ID FROM Genre WHERE Name = 'Mystery & Thriller'));

-- Insert new author
INSERT INTO Author (author_id, Name)
VALUES (nextval('author_sequence'),'Lucas Luke');

-- Insert authorship record
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
VALUES((nextval('authorship_sequence')),(SELECT Author_ID FROM Author
WHERE Name = 'Lucas Luke'), (SELECT CURRVAL('material_sequence')));
COMMIT;
```

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

```sql
-- Insert new material
INSERT INTO Material (material_id, Title, Publication_Date, Catalog_ID, Genre_ID)
VALUES (nextval('material_sequence'),'New Book', '2020-08-01',
        (SELECT Catalog_ID FROM Catalog WHERE Name = 'E-Books'),
        (SELECT Genre_ID FROM Genre WHERE Name = 'Mystery & Thriller'));

select * from material m where title = 'New Book';
-- Insert new author
INSERT INTO Author (author_id, Name)
VALUES (nextval('author_sequence'),'Lucas Luke');

-- Insert authorship record
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
VALUES((nextval('authorship_sequence')),(SELECT Author_ID FROM Author WHERE Name = 'Lucas Luke'),
(SELECT CURRVAL('material_sequence')));
```

material 1

select * from material m where title = 'I | Enter a SQL expression to filter results (use Ctrl+Space)

	material_id	title	publication_date	catalog_id	genre_id
1	32	New Book	2020-08-01	3	2

```sql
-- Insert new author
INSERT INTO Author (author_id, Name)
VALUES (nextval('author_sequence'),'Lucas Luke');

select * from Author where name = 'Lucas Luke';

-- Insert authorship record
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
```

author 1

select * from Author where name = 'Lu | Enter a SQL expression to filter results (us

	author_id	name	birth_date	nationality
1	21	Lucas Luke	[NULL]	[NULL]

```sql
-- Insert authorship record
INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID)
VALUES((nextval('authorship_sequence')),(SELECT Author_ID FROM Author WHERE Name = 'Lucas Luke'),
(SELECT CURRVAL('material_sequence')));

select * from authorship where material_id = 32;
```

authorship 1

select * from authorship where materia | Enter a SQL expression to filter results (use Ctrl+Space)

	authorship_id	author_id	material_id
1	35	21	32

MUKESH RAJMOHAN – G01456275
PRANAVI RAO LINGALA - G01448232

CS 504 - PROJECT

DESIGN:

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

- A simple SQL query can be written to select overdue materials by joining the Borrow and Material tables:

```sql
SELECT m.title, b.borrow_date, b.due_date
FROM Borrow b JOIN Material m
ON b.material_id = m.material_id
where b.return_date IS NULL and b.due_date < CURRENT_DATE;
```

- This can be scheduled as a batch script to run daily.

- The results could be written to a notification table that staff can query. Or can be sent via email/API call.

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.

- Create a table to track overdue counts:

```sql
CREATE TABLE member_overdues (
      member_id INTEGER PRIMARY KEY,
      overdue_count INTEGER DEFAULT 0
);
```

- Use a scheduled SQL script that:
  - Joins Borrow and Member tables
  - Sums overdue for each member
  - Updates member_overdue table with overdue counts
  - Updates Member table setting active=False where overdue_count >= 3

```sql
-- Calculate overdue counts
INSERT INTO member_overdues
SELECT member_id, COUNT(*) as overdue_count
FROM Borrow b JOIN Member m ON b.member_id = m.member_id
WHERE b.return_date IS NULL
GROUP BY member_id;

-- Deactivate members
UPDATE Member m
SET active = FALSE
WHERE EXISTS (
      SELECT 1 FROM member_overdues
      WHERE member_id = m.member_id AND overdue_count >= 3);
```

CS 504 - PROJECT

- On Overdue fee payment:

```
UPDATE member_overdues
SET overdue_count = 0
WHERE member_id = :member_id;

UPDATE Member
SET active = TRUE
WHERE member_id = :member_id;
```