# 2024 Spring CS504 Principles of Data Management and Mining Project Report

Pramath Rajprasad Rao

G01483865

---

## Introduction

The library helps people in finding information and learning. Library will have a vast set of genres which has to be managed by the librarians. To look after these books is very difficult where the librarians must maintain a manual record of each member. To overcome this library management was introduced it makes tasks simplified. It tracks all the members, due dates, return dates, overdue, and many other tasks. It makes helps the librarians and provides a better service to their members.

**2. Database Design**

**1. Define the scope of the project and identify the entities and their relationships.**

**Scope of the project**

This project's focus is on building a database management system for the library. This system helps the librarian to manage their resources which includes different books and genres. It will also help in locating the books, providing detailed information for their members, and monitoring the borrowing and their dues. It provides better services makes it easier for both staff and members and should maintain data integrity and reduce redundancy.

Features:

Material Management:

All the library materials including books, magazines, e-books, and audiobooks, should be stored and maintained in the system together with information on their titles, authors, publication dates, and categories.

Membership Management:

The information regarding library users, such as their names, contact details, membership numbers, and borrowing histories, should be stored and managed by the system.

Borrowing:

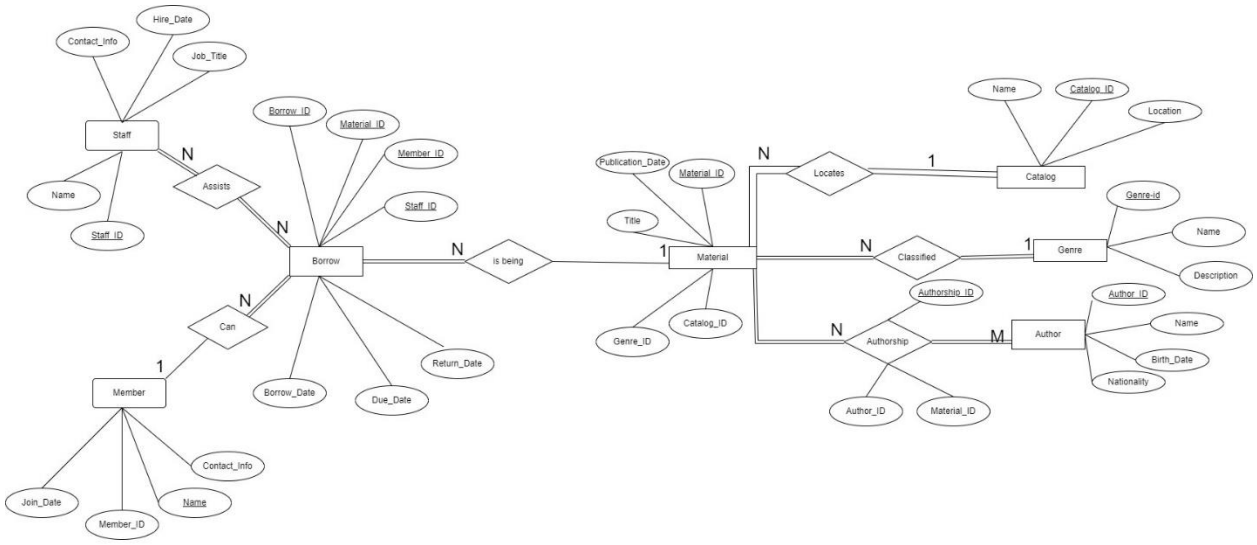
The system needs to streamline the process of borrowing materials, enable members to check them out and give library employees the data they need to oversee the circulation of library materials. Following a material's checkout, a librarian ought to note

its anticipated due date or borrow date. Additionally, the material's return date needs to be changed after it is returned.

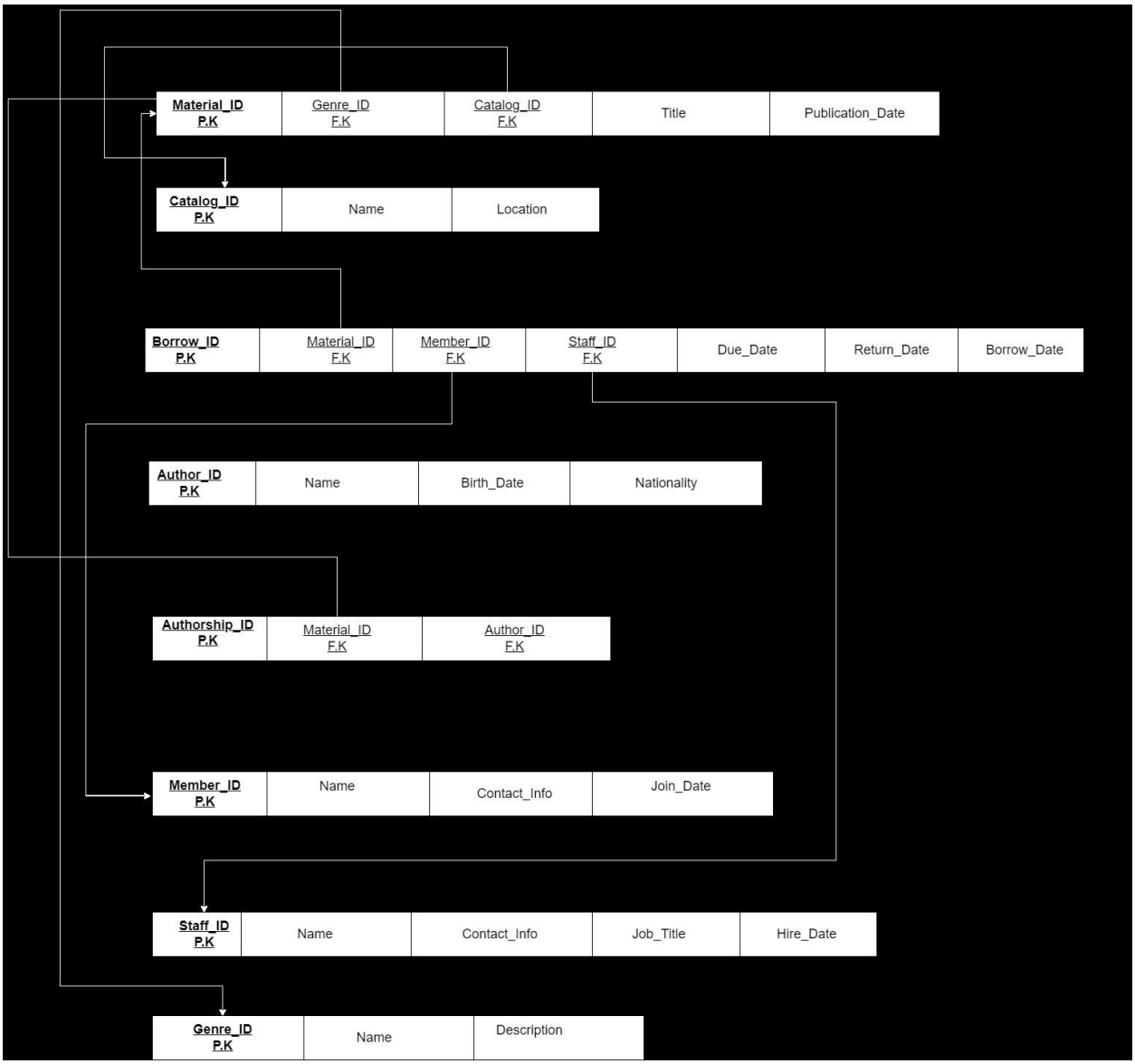
Reporting and Analytics:

The system should produce reports on popular materials, library usage, and other pertinent statistics so that library employees may make data-driven choices on the acquisition and management of resources.

**Entity-Relationship Diagram:**



Relational Schema:




Entites:

Materials: This contains from the library such as books, magazines, e-books, and audiobooks.

Catalog: This contains the details of the books and their location if they are available.

Genre: This contains different categories of books in the library.

Author: This is the information of the person who has written the book and their details.

Member: The people who are the members of the library and borrow books.

Staff: The people who work at the library where they manage all the resources and provide good experience.

Many to one:

Material and Genre entities are **classified** relationships in which Genre and Material are totally participating.

Material and Catalog entities **locate** a relationship in which Material and Catalog are totally participating.

Material and Borrow entities **is belong** relationship in which Material is partially participating and Borrow is totally participating.

Material and Members **can** have a relationship in which Borrow is totally participating and member is partially participating.

Many to Many:

Author and Material entities are in **authorship** and are totally participating with each other.

Staff and Borrow entities **assist** in the relationship and are totally participating.

**3.Database Implementation**

**1. . Choose and install an appropriate Database Management System (DBMS) for this project.**

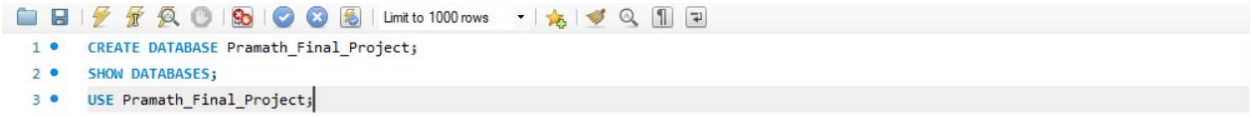
I have chosen My SQL Workbench for creating database, and tables, and inserting the values into them. This application offered an easy-to-use interface for creating tables, establishing the database schema, and setting up relationships between them. I could construct and manage the database structure visually with MySQL Workbench, which made the development process more structured and productive.

**DDL**

**Creating a database**

**SQL Command:**

```
CREATE DATABASE Pramath_Final_Project;
SHOW DATABASES;
USE Pramath_Final_Project;
```

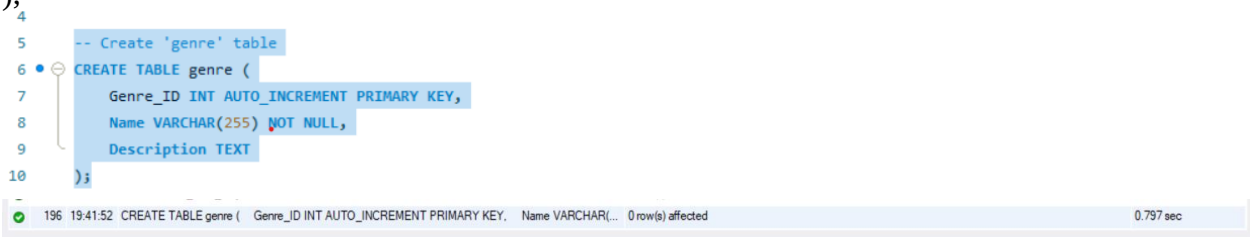




## Creating tables

### Genre table

```
CREATE TABLE genre (
   Genre_ID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
   Description TEXT
);
```




### Catalog table

```
CREATE TABLE catalog (
        Catalog_ID INT AUTO_INCREMENT PRIMARY KEY,
   Name VARCHAR(255) NOT NULL,
   Location VARCHAR(255) NOT NULL
);
```

```sql
-- Create 'Catalog' table
CREATE TABLE catalog (
    Catalog_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Location VARCHAR(255) NOT NULL
);
```

197 19:44:06 CREATE TABLE catalog ( Catalog_ID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR... 0 row(s) affected 0.266 sec

**Material Table**

```sql
CREATE TABLE material (
        Material_ID INT AUTO_INCREMENT PRIMARY KEY,
  Title VARCHAR(255) NOT NULL,
  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)
);
```

```sql
-- Create 'Material' table
CREATE TABLE material (
    Material_ID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    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)
);
```

198 19:46:19 CREATE TABLE material ( Material_ID INT AUTO_INCREMENT PRIMARY KEY, Title VARCHAR(... 0 row(s) affected 0.204 sec

**Author Table**

```sql
CREATE TABLE author (
        Author_ID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Birth_Date Date,
  Nationality VARCHAR(100)
  );
```

```sql
-- Create 'Author' table
CREATE TABLE author (
    Author_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255) NOT NULL,
    Birth_Date Date,
    Nationality VARCHAR(100)
    );
```

199 19:48:09 CREATE TABLE author ( Author_ID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(2... 0 row(s) affected 0.110 sec

**Authorship Table**

```sql
CREATE TABLE authorship (
        Authorship_ID INT AUTO_INCREMENT PRIMARY KEY,
  Author_ID INT,
  Material_ID INT,
  FOREIGN KEY (Author_ID) REFERENCES author(Author_ID),
  FOREIGN KEY (Material_ID) REFERENCES material(Material_ID)
  );
```

```sql
-- Create 'Authorship' table
CREATE TABLE authorship (
    Authorship_ID INT AUTO_INCREMENT PRIMARY KEY,
    Author_ID INT,
    Material_ID INT,
    FOREIGN KEY (Author_ID) REFERENCES author(Author_ID),
    FOREIGN KEY (Material_ID) REFERENCES material(Material_ID)
    );
```

200 19:50:51 CREATE TABLE authorship ( Authorship_ID INT AUTO_INCREMENT PRIMARY KEY, Author_ID I... 0 row(s) affected 0.469 sec

**Member Table**

```sql
CREATE TABLE member (
        Member_ID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Contact_Info VARCHAR(100),
  Join_Date DATE
  );
```

```
47      -- Create 'Member' table
48  CREATE TABLE member (
49      Member_ID INT AUTO_INCREMENT PRIMARY KEY,
50      Name VARCHAR(100) NOT NULL,
51      Contact_Info VARCHAR(100),
52      Join_Date DATE
53      );
```

201 19:57:10 CREATE TABLE member ( Member_ID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHA... 0 row(s) affected 0.485 sec

**Staff Table**

```
CREATE TABLE staff (
        Staff_ID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(100) NOT NULL,
  Contact_Info VARCHAR(100),
  Job_Title VARCHAR(100),
  Hire_Date DATE
  );
```

```
55      -- Create 'Staff' table
56  CREATE TABLE staff (
57      Staff_ID INT AUTO_INCREMENT PRIMARY KEY,
58      Name VARCHAR(100) NOT NULL,
59      Contact_Info VARCHAR(100),
60      Job_Title VARCHAR(100),
61      Hire_Date DATE
62      );
```

3 20:01:21 CREATE TABLE staff ( Staff_ID INT AUTO_INCREMENT PRIMARY KEY, Name VARCHAR(100) NO... 0 row(s) affected 1.187 sec

**Borrow Table**

```
CREATE TABLE borrow (
  Borrow_ID INT AUTO_INCREMENT PRIMARY KEY,
  Material_ID INT,
  Member_ID INT,
  Staff_ID INT,
  Borrow_Date DATE NOT NULL,
  Due_Date DATE NOT NULL,
  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)
);
```

```
-- Create 'borrow' table
CREATE TABLE borrow (
    Borrow_ID INT AUTO_INCREMENT PRIMARY KEY,
    Material_ID INT,
    Member_ID INT,
    Staff_ID INT,
    Borrow_Date DATE NOT NULL,
    Due_Date DATE NOT NULL,
    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)
);
```

4 20:03:45 CREATE TABLE borrow ( Borrow_ID INT AUTO_INCREMENT PRIMARY KEY, Material_ID INT, ... 0 row(s) affected 0.203 sec

Displaying all the tables.

SHOW TABLES;

Limit to 1000 rows

```
    Due_Date DATE NOT NULL,
    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)
);

SHOW TABLES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Tables_in_pramath_final_project |
|---|
| author |
| authorship |
| borrow |
| catalog |
| genre |
| material |
| member |
| staff |

Result Grid | Form Editor | Field Types

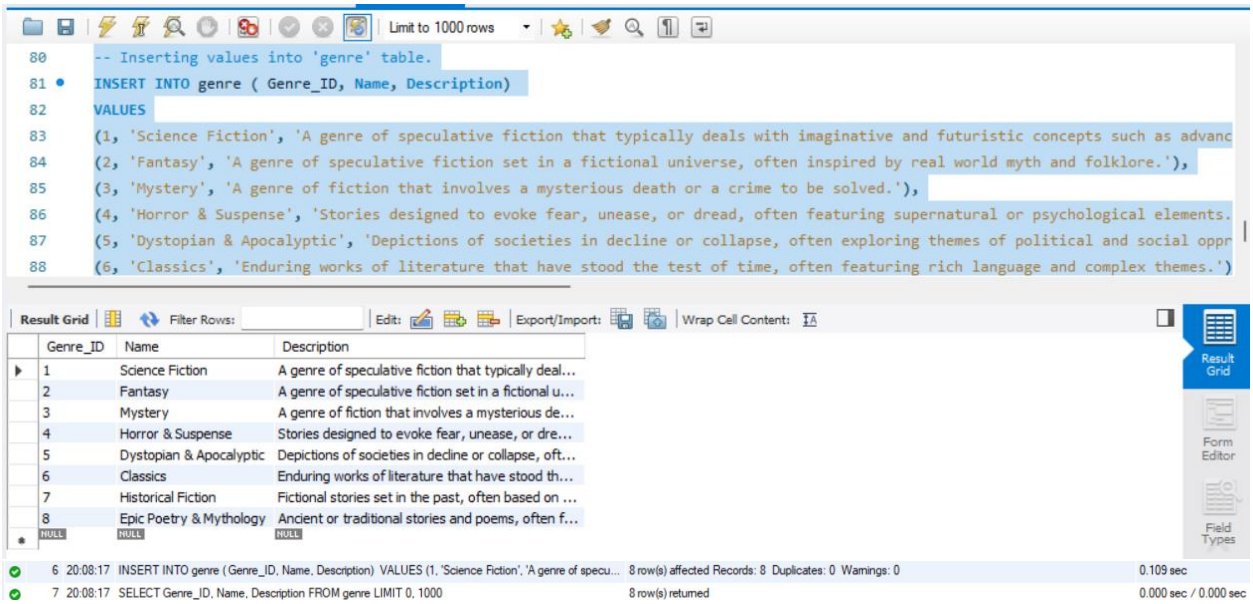
5 20:04:54 SHOW TABLES 8 row(s) returned 0.000 sec / 0.000 sec

**Inserting values into the table.**

**Genre Table**

-- Inserting values into 'genre' table.
INSERT INTO genre ( Genre_ID, Name, Description)
VALUES

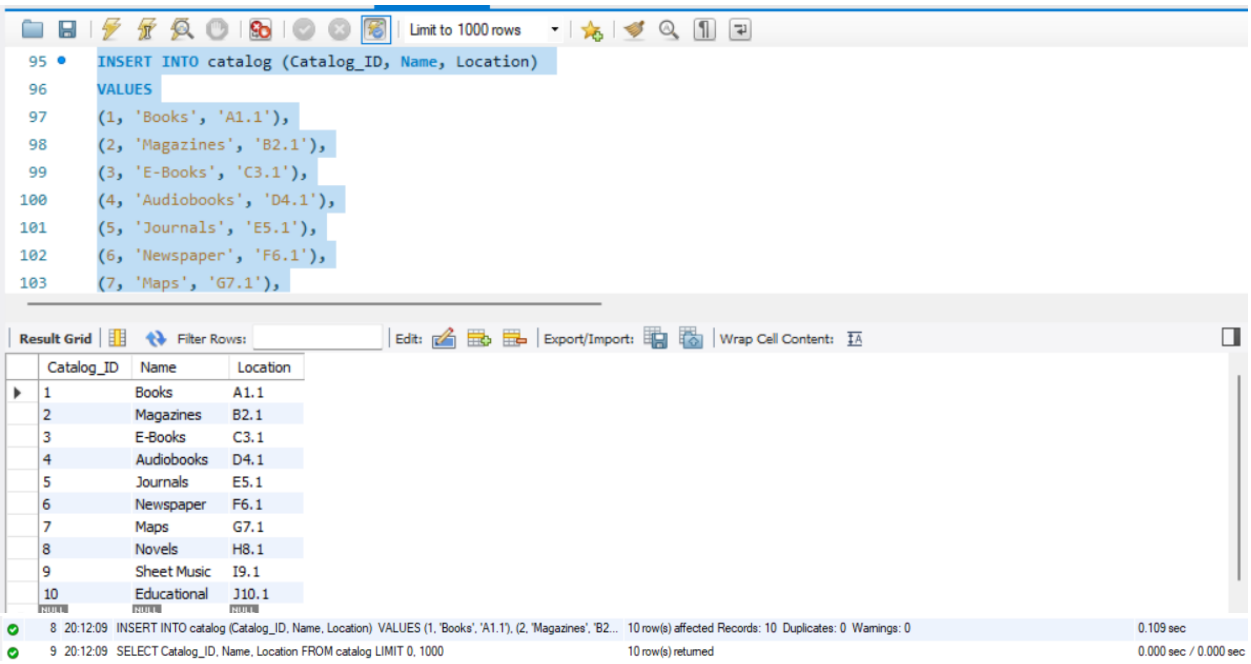
(1, 'Science Fiction', 'A genre of speculative fiction that typically deals with imaginative and futuristic concepts such as advanced science and technology, space exploration, time travel, parallel universes, and extraterrestrial life.'),
(2, 'Fantasy', 'A genre of speculative fiction set in a fictional universe, often inspired by real world myth and folklore.'),
(3, 'Mystery', 'A genre of fiction that involves a mysterious death or a crime to be solved.'),
(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.');
-- This is to print the values in genre table.
SELECT Genre_ID, Name, Description FROM genre;




**Catalog Table**

INSERT INTO catalog (Catalog_ID, Name, Location)
VALUES
(1, 'Books', 'A1.1'),
(2, 'Magazines', 'B2.1'),

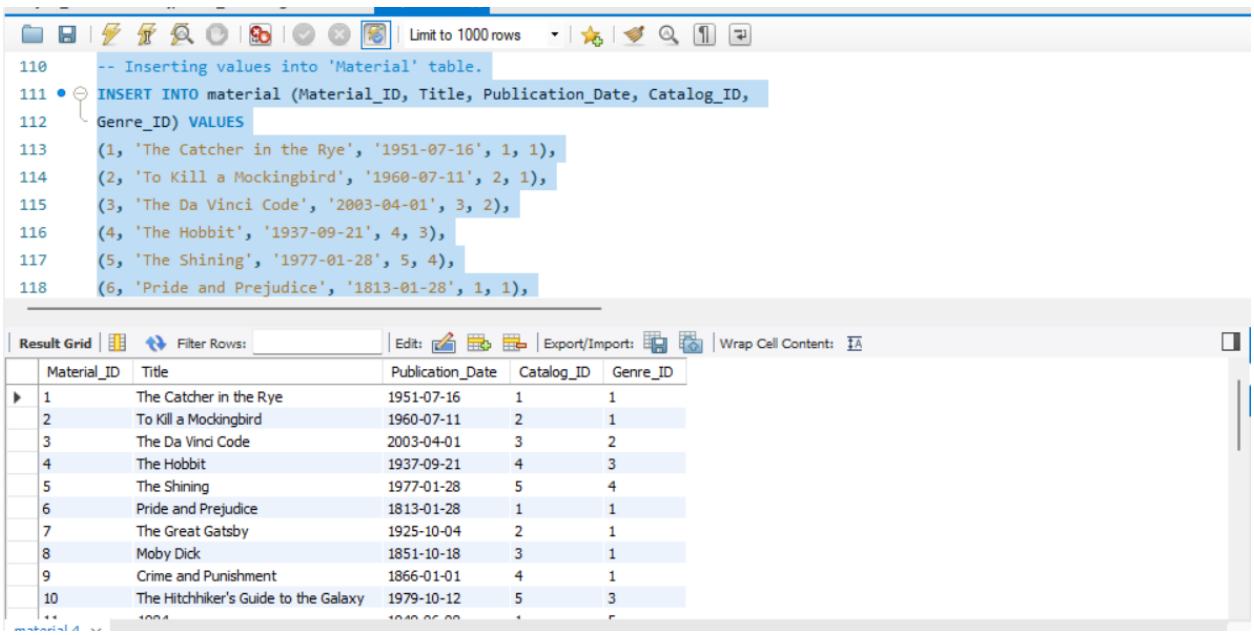
(3, 'E-Books', 'C3.1'),
(4, 'Audiobooks', 'D4.1'),
(5, 'Journals', 'E5.1'),
(6, 'Newspaper', 'F6.1'),
(7, 'Maps', 'G7.1'),
(8, 'Novels', 'H8.1'),
(9, 'Sheet Music', 'I9.1'),
(10, 'Educational', 'J10.1');
-- To print the values from catalog table.
SELECT Catalog_ID, Name, Location FROM catalog;




**Material Table**

-- Inserting values into 'Material' table.
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', '1925-10-04', 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, 'The 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 Night''s Dream', '1596-01-01', 10, 7),
(31, 'The Tricky Book', '1888-01-01', 10, 7);
-- Print the material table.
SELECT * FROM material;

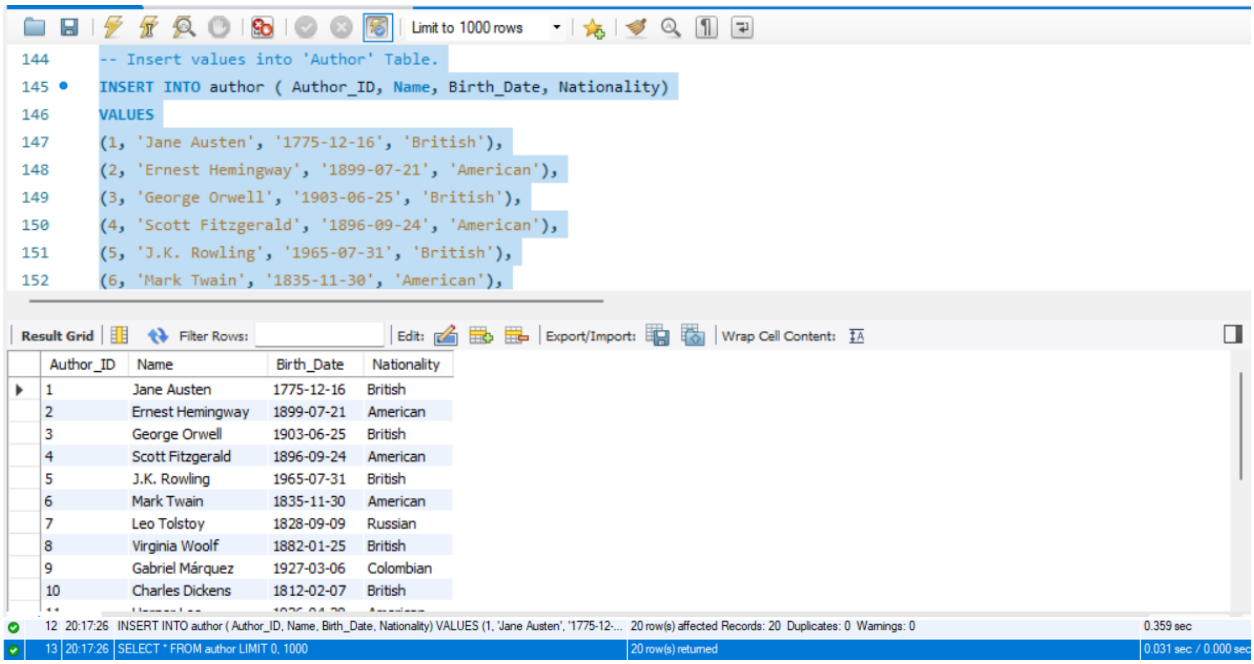



| Material_ID | Title | Publication_Date | Catalog_ID | Genre_ID |
|---|---|---|---|---|
| 1 | The Catcher in the Rye | 1951-07-16 | 1 | 1 |
| 2 | To Kill a Mockingbird | 1960-07-11 | 2 | 1 |
| 3 | The Da Vinci Code | 2003-04-01 | 3 | 2 |
| 4 | The Hobbit | 1937-09-21 | 4 | 3 |
| 5 | The Shining | 1977-01-28 | 5 | 4 |
| 6 | Pride and Prejudice | 1813-01-28 | 1 | 1 |
| 7 | The Great Gatsby | 1925-10-04 | 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 |

| | | | | | |
|---|---|---|---|---|---|
| 10 | 20:14:15 | INSERT INTO material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID) VALUES (1, 'The... | 31 row(s) affected Records: 31 Duplicates: 0 Warnings: 0 | | 0.015 sec |
| 11 | 20:14:15 | SELECT * FROM material LIMIT 0, 1000 | 31 row(s) returned | | 0.000 sec / 0.000 sec |

**Author Table**

```
-- Insert values into 'Author' Table.
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');
-- Print the 'Author' Table
SELECT * FROM author;
```




**Authorship Table**

-- Insert values into 'authorship' Table
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),(23,3,22),(24,3,23),(25,4,24),(26,5,25),
(27,6,26),(28,7,27),(29,8,28),(30,19,28),(31,9,29),(32,10,30),(33,8,30),
(34,2,29);
-- Print the values from 'authorship' table
SELECT Authorship_ID, Author_ID, Material_ID FROM 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),(23,3,22),(24,3,23),(25,4,24),(26,5,25),
(27,6,26),(28,7,27),(29,8,28),(30,19,28),(31,9,29),(32,10,30),(33,8,30),
(34,2,29);
-- Print the values from 'authorship' table
SELECT Authorship_ID, Author_ID, Material_ID FROM authorship;
```

| Authorship_ID | Author_ID | Material_ID |
|---|---|---|
| 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 |

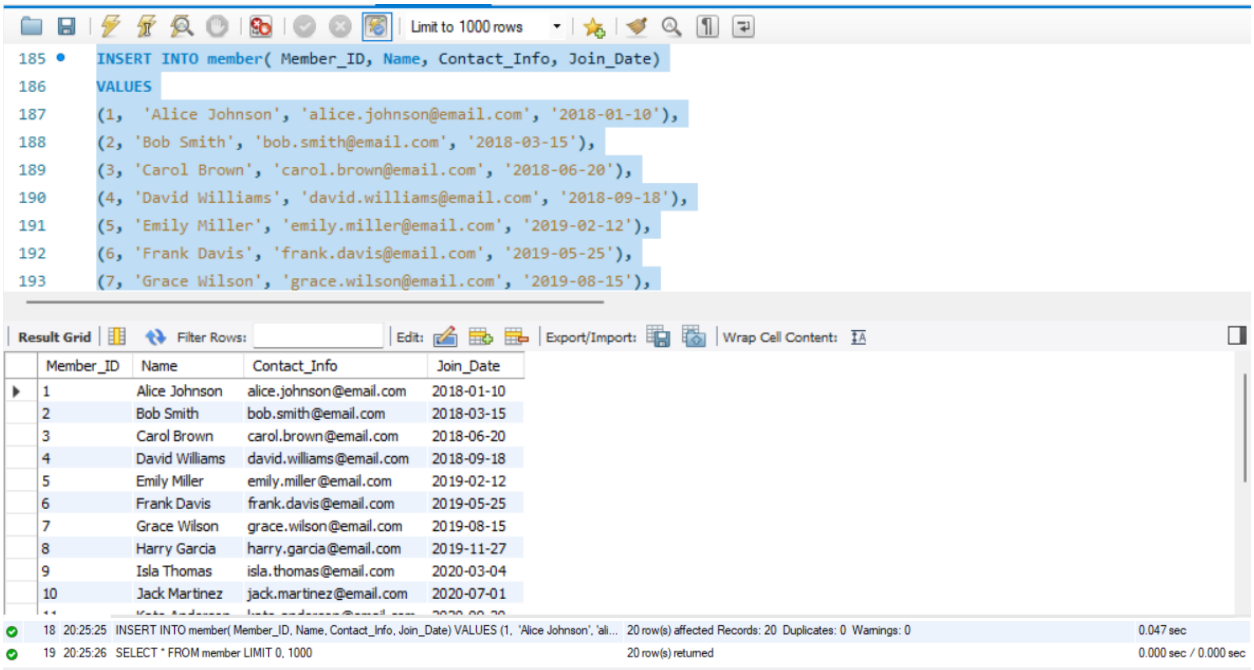
16 20:22:17 INSERT INTO authorship ( Authorship_ID, Author_ID, Material_ID) VALUES (1,1,1),(2,2,2),(3,3,3),(4,4,... 34 row(s) affected Records: 34 Duplicates: 0 Warnings: 0 0.141 sec

17 20:22:18 SELECT Authorship_ID, Author_ID, Material_ID FROM authorship LIMIT 0, 1000 34 row(s) returned 0.000 sec / 0.000 sec

**Member Table**

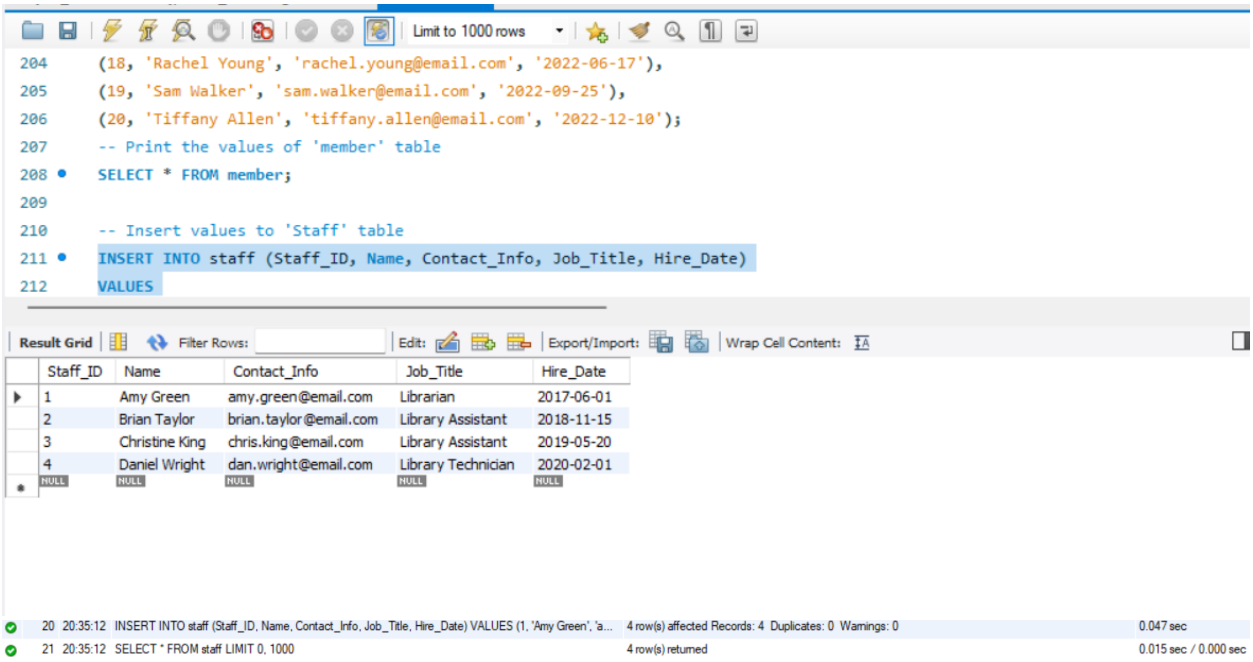
```
-- Insert values into 'member' table
INSERT INTO member( Member_ID, Name, Contact_Info, Join_Date)
VALUES
(1,  'Alice Johnson', 'alice.johnson@email.com', '2018-01-10'),
(2, 'Bob Smith', 'bob.smith@email.com', '2018-03-15'),
(3, 'Carol Brown', 'carol.brown@email.com', '2018-06-20'),
(4, 'David Williams', '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');
-- Print the values of 'member' table
SELECT * FROM member;




**Staff Table**

-- Insert values to 'Staff' table
INSERT INTO staff (Staff_ID, 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');
-- Print the values of 'staff' table
SELECT * FROM staff;



```
(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');
-- Print the values of 'member' table
SELECT * FROM member;

-- Insert values to 'Staff' table
INSERT INTO staff (Staff_ID, Name, Contact_Info, Job_Title, Hire_Date)
VALUES
```

| Staff_ID | Name | Contact_Info | Job_Title | Hire_Date |
|---|---|---|---|---|
| 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 |

```
20 20:35:12 INSERT INTO staff (Staff_ID, Name, Contact_Info, Job_Title, Hire_Date) VALUES (1, 'Amy Green', 'a... 4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0 0.047 sec
21 20:35:12 SELECT * FROM staff LIMIT 0, 1000 4 row(s) returned 0.015 sec / 0.000 sec
```

**Borrow Table**

```
-- Insert values into 'borrow' table
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', NULL),
(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);
-- Print the values of 'borrow' table
SELECT * FROM borrow;
```

Limit to 1000 rows

```
221 • INSERT INTO borrow ( Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)
222   VALUES
223   (1, 1, 1, 1, '2018-09-12', '2018-10-03', '2018-09-30'),
224   (2, 2, 2, 1, '2018-10-15', '2018-11-05', '2018-10-29'),
225   (3, 3, 3, 1, '2018-12-20', '2019-01-10', '2019-01-08'),
226   (4, 4, 4, 1, '2019-03-11', '2019-04-01', '2019-03-27'),
227   (5, 5, 5, 1, '2019-04-20', '2019-05-11', '2019-05-05'),
228   (6, 6, 6, 1, '2019-07-05', '2019-07-26', '2019-07-21'),
229   (7, 7, 7, 1, '2019-09-10', '2019-10-01', '2019-09-25'),
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| Borrow_ID | Material_ID | Member_ID | Staff_ID | Borrow_Date | Due_Date | Return_Date |
|---|---|---|---|---|---|---|
| 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 |

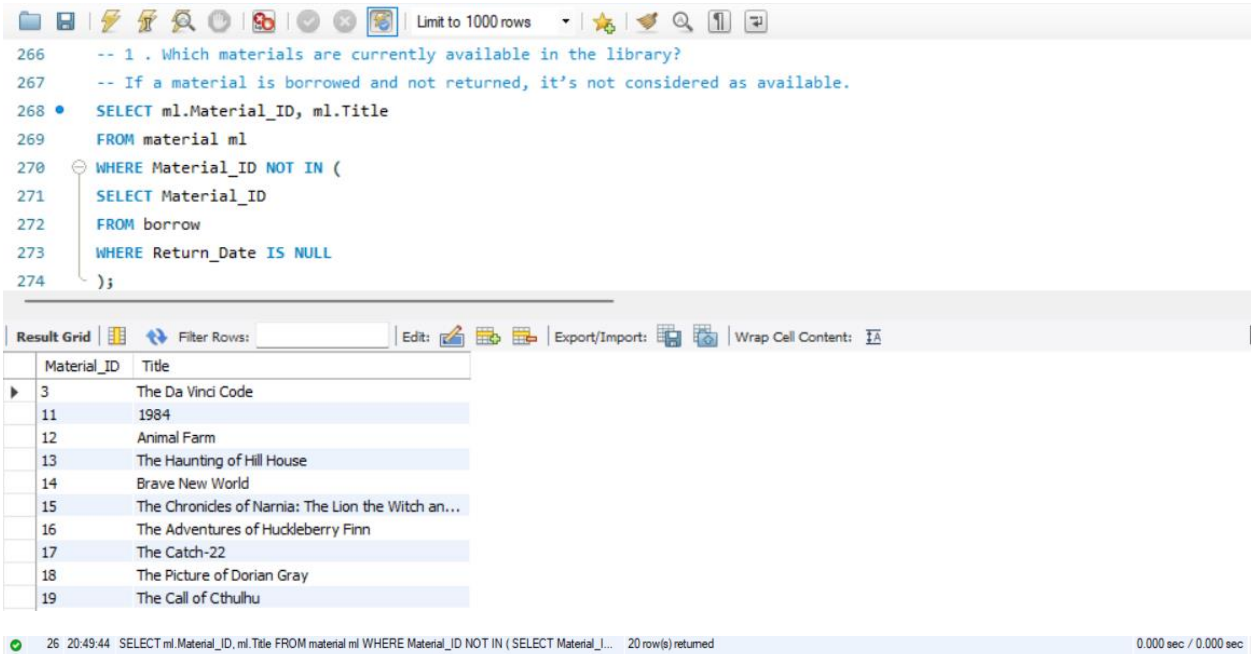
22 20:38:54 INSERT INTO borrow ( Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Retu... 40 row(s) affected Records: 40 Duplicates: 0 Warnings: 0 1.171 sec

23 20:38:55 SELECT * FROM borrow LIMIT 0, 1000 40 row(s) returned 0.000 sec / 0.000 sec

**4.Querying and Manipulation**

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

```
-- 1 . Which materials are currently available in the library?
-- If a material is borrowed and not returned, it's not considered as available.
SELECT ml.Material_ID, ml.Title
FROM material ml
WHERE Material_ID NOT IN (
SELECT Material_ID
FROM borrow
WHERE Return_Date IS NULL
);
```




| Material_ID | Title |
|---|---|
| 3 | The Da Vinci Code |
| 11 | 1984 |
| 12 | Animal Farm |
| 13 | The Haunting of Hill House |
| 14 | Brave New World |
| 15 | The Chronicles of Narnia: The Lion the Witch an... |
| 16 | The Adventures of Huckleberry Finn |
| 17 | The Catch-22 |
| 18 | The Picture of Dorian Gray |
| 19 | The Call of Cthulhu |

**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 04/01/2023, and show the borrow date and due date of each material.
SELECT ml.Material_ID, ml.Title, bw.Borrow_Date, bw.Due_Date
FROM material ml
JOIN borrow bw ON ml.Material_ID = bw.Material_ID
WHERE bw.Due_Date < '2023-04-01' AND bw.Return_Date IS NULL
ORDER BY ml.Material_ID;
```

```
-- 2. Which materials are currently overdue?
-- Suppose today is 04/01/2023, and show the borrow date and due date of each material.
SELECT ml.Material_ID, ml.Title, bw.Borrow_Date, bw.Due_Date
FROM material ml
JOIN borrow bw ON ml.Material_ID = bw.Material_ID
WHERE bw.Due_Date < '2023-04-01' AND bw.Return_Date IS NULL
ORDER BY ml.Material_ID;
```

Result Grid

| Material_ID | Title | Borrow_Date | Due_Date |
|---|---|---|---|
| 1 | The Catcher in the Rye | 2022-12-28 | 2023-01-18 |
| 2 | 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 |
| 20 | Harry Potter and the Philosopher's Stone | 2021-10-21 | 2021-11-11 |
| 21 | Frankenstein | 2021-11-29 | 2021-12-20 |

27 20:55:31 SELECT ml.Material_ID, ml.Title, bw.Borrow_Date, bw.Due_Date FROM material ml JOIN borrow bw O... 6 row(s) returned 0.000 sec / 0.000 sec

**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**

```
-- 3. What are the top 10 most borrowed materials in the library?
-- Show the <tle of each material and order them based on their available counts.
SELECT ml.Material_ID, ml.Title, COUNT(bw.Material_ID) AS Count_Borrow
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
WHERE Return_Date IS NOT NULL
GROUP BY bw.Material_ID, ml.Title
ORDER BY Count_Borrow DESC
LIMIT 10;
```

```sql
-- 3. What are the top 10 most borrowed materials in the library?
-- Show the <tle of each material and order them based on their available counts.
SELECT ml.Material_ID, ml.Title, COUNT(bw.Material_ID) AS Count_Borrow
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
WHERE Return_Date IS NOT NULL
GROUP BY bw.Material_ID, ml.Title
ORDER BY Count_Borrow DESC
LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

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

30 21:06:09 SELECT ml.Material_ID, ml.Title, COUNT(bw.Material_ID) AS Count_Borrow FROM borrow bw JOIN m... 10 row(s) returned 0.031 sec / 0.000 sec

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

```sql
-- 4. How many materials has the author Lucas Piki written?
SELECT ar.Name,COUNT(ml.Material_ID) AS Total_Author_Materials
FROM author ar
JOIN authorship aup ON ar.Author_ID = aup.Author_ID
JOIN material ml ON aup.Material_ID = ml.Material_ID
WHERE ar.Name = 'Lucas Piki';
```

Project_Code* | Rajprasad_RaoAssignment-5 | SQL File 4*

Limit to 1000 rows

```sql
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
WHERE Return_Date IS NOT NULL
GROUP BY bw.Material_ID, ml.Title
ORDER BY Count_Borrow DESC
LIMIT 10;

-- 4. How many materials has the author Lucas Piki written?
SELECT ar.Name,COUNT(ml.Material_ID) AS Total_Author_Materials
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Name | Total_Author_Materials |
|---|---|
| Lucas Piki | 1 |

31 21:08:13 SELECT ar.Name,COUNT(ml.Material_ID) AS Total_Author_Materials FROM author ar JOIN authorshi... 1 row(s) returned 0.344 sec / 0.000 sec

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

```
-- 5. How many materials were written by two or more authors?
SELECT COUNT(*) AS Materials_With_Multiple_Authors
FROM (
   SELECT Material_ID
   FROM authorship
   GROUP BY Material_ID
   HAVING COUNT(Author_ID) >=2
) AS subquery;
```

Limit to 1000 rows

```
288     JOIN material ml ON bw.Material_ID = ml.Material_ID
289     WHERE Return_Date IS NOT NULL
290     GROUP BY bw.Material_ID, ml.Title
291     ORDER BY Count_Borrow DESC
292     LIMIT 10;
293
294
295     -- 5. How many materials were written by two or more authors?
296     SELECT COUNT(*) AS Materials_With_Multiple_Authors
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Materials_With_Multiple_Authors |
|---|
| 4 |

32 21:10:33 SELECT COUNT(*) AS Materials_With_Multiple_Authors FROM ( SELECT Material_ID FROM aut... 1 row(s) returned 0.000 sec / 0.000 sec

```
SELECT ml.material_ID,ml.Title, COUNT(DISTINCT ar.Author_ID) AS Author_Count
FROM material ml
JOIN authorship aup ON ml.Material_ID = aup.Material_ID
JOIN author ar ON ar.Author_ID = aup.Author_ID
GROUP BY ml.Material_ID, ml.Title
ORDER BY Author_Count DESC
LIMIT 4;
```

```
SELECT ml.material_ID,ml.Title, COUNT(DISTINCT ar.Author_ID) AS Author_Count
FROM material ml
JOIN authorship aup ON ml.Material_ID = aup.Material_ID
JOIN author ar ON ar.Author_ID = aup.Author_ID
GROUP BY ml.Material_ID, ml.Title
ORDER BY Author_Count DESC
LIMIT 4;
```

Result Grid

| material_ID | Title | Author_Count |
|---|---|---|
| 29 | The Count of Monte Cristo | 2 |
| 30 | A Midsummer Night's Dream | 2 |
| 28 | The Old Man and the Sea | 2 |
| 22 | A Tale of Two Cities | 2 |

33 21:11:34 SELECT ml.material_ID,ml.Title, COUNT(DISTINCT ar.Author_ID) AS Author_Count FROM material ml ... 4 row(s) returned 0.016 sec / 0.000 sec

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

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

```
SELECT ge.Name AS Genre_Name, COUNT(bw.Borrow_ID) AS Count_Borrows
FROM genre ge
JOIN material ml ON ge.Genre_ID = ml.Genre_ID
JOIN borrow bw ON ml.Material_ID = bw.Material_ID
GROUP BY ge.Genre_ID, ge.Name
ORDER BY Count_Borrows DESC;
```

```
-- 6. What are the most popular genres in the library ranked by the total number of borrowed times of each genre?
SELECT ge.Name AS Genre_Name, COUNT(bw.Borrow_ID) AS Count_Borrows
FROM genre ge
JOIN material ml ON ge.Genre_ID = ml.Genre_ID
JOIN borrow bw ON ml.Material_ID = bw.Material_ID
GROUP BY ge.Genre_ID, ge.Name
ORDER BY Count_Borrows DESC;
```

| Genre_Name | Count_Borrows |
|---|---|
| Science Fiction | 22 |
| Mystery | 6 |
| Horror & Suspense | 5 |
| Fantasy | 3 |
| Classics | 3 |
| Historical Fiction | 1 |

35 21:15:43 SELECT ge.Name AS Genre_Name, COUNT(bw.Borrow_ID) AS Count_Borrows FROM genre ge JOI... 6 row(s) returned 0.031 sec / 0.000 sec

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

```
-- 7. How many materials had been borrowed from 09/2020-10/2020?
SELECT ml.Title, COUNT(*) AS Materials_Borrowed
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
WHERE bw.Borrow_Date >= '2020-09-01' AND bw.Borrow_Date <= '2020-10-31'
GROUP BY ml.Title
ORDER BY Materials_Borrowed DESC;
```

```
-- 7. How many materials had been borrowed from 09/2020-10/2020?
SELECT ml.Title, COUNT(*) AS Materials_Borrowed
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
WHERE bw.Borrow_Date >= '2020-09-01' AND bw.Borrow_Date <= '2020-10-31'
GROUP BY ml.Title
ORDER BY Materials_Borrowed DESC;
```

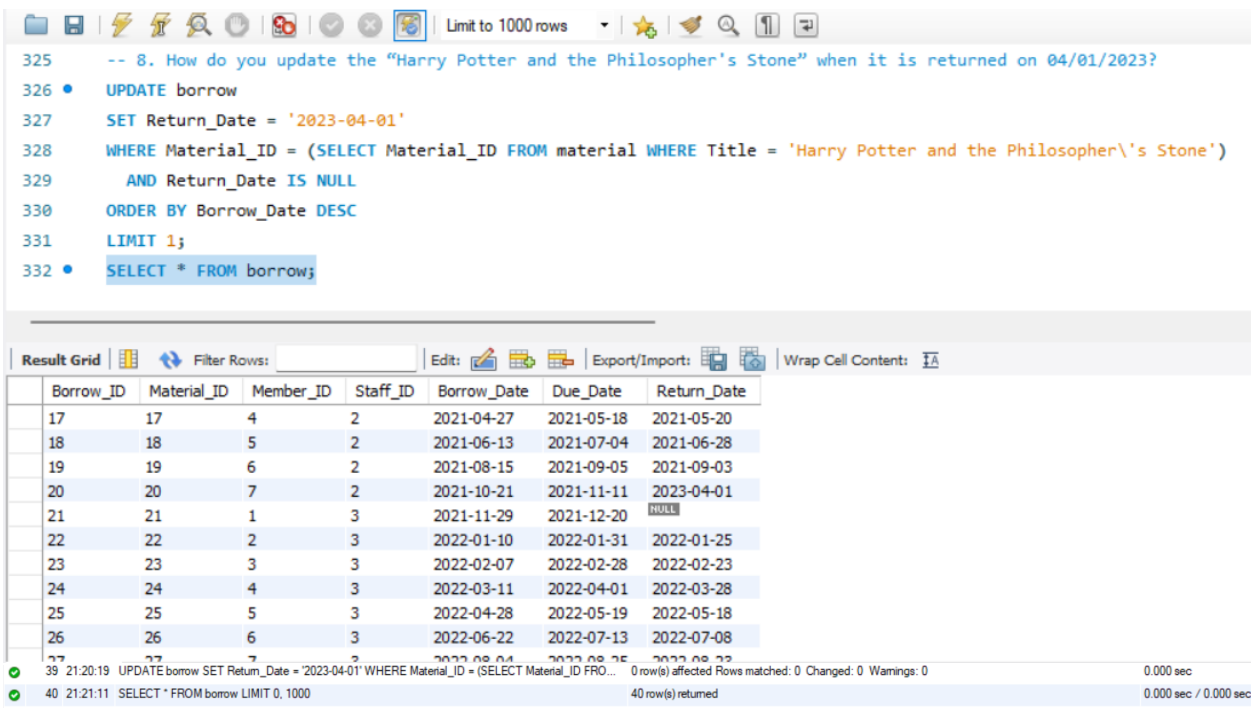
| Title | Materials_Borrowed |
|---|---|
| The Da Vinci Code | 1 |

36 21:17:29 SELECT ml.Title, COUNT(*) AS Materials_Borrowed FROM borrow bw JOIN material ml ON bw.Materia... 1 row(s) returned 0.016 sec / 0.000 sec

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

-- 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 = (SELECT Material_ID FROM material WHERE Title = 'Harry Potter and the Philosopher\'s Stone')
  AND Return_Date IS NULL
ORDER BY Borrow_Date DESC
LIMIT 1;
SELECT * FROM borrow;
```




| Borrow_ID | Material_ID | Member_ID | Staff_ID | Borrow_Date | Due_Date | Return_Date |
|---|---|---|---|---|---|---|
| 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 | 2023-04-01 |
| 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 |

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

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

```
DELETE FROM borrow
WHERE Member_ID = (
   SELECT Member_ID FROM member WHERE Name = 'Emily Miller'
);
SELECT * FROM borrow WHERE Member_ID = 5;
```

```
SELECT * FROM borrow;

-- 9. How do you delete the member Emily Miller and all her related records from the database?
DELETE FROM borrow
WHERE Member_ID = (
    SELECT Member_ID FROM member WHERE Name = 'Emily Miller'
);
SELECT * FROM borrow WHERE Member_ID = 5;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| Borrow_ID | Material_ID | Member_ID | Staff_ID | Borrow_Date | Due_Date | Return_Date |
|---|---|---|---|---|---|---|
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

| | | | | |
|---|---|---|---|---|
| 41 | 21:24:10 | DELETE FROM borrow WHERE Member_ID = ( SELECT Member_ID FROM member WHERE Na... | 3 row(s) affected | 0.235 sec |
| 42 | 21:25:55 | SELECT * FROM borrow WHERE Member_ID = 5 LIMIT 0, 1000 | 0 row(s) returned | 0.000 sec / 0.000 sec |

DELETE FROM member
WHERE Member_ID IN (
  SELECT Member_ID FROM (SELECT Member_ID FROM member WHERE Name = 'Emily Miller' LIMIT 1) AS subquery
);

SELECT * FROM member;

```
DELETE FROM member
WHERE Member_ID IN (
    SELECT Member_ID FROM (SELECT Member_ID FROM member WHERE Name = 'Emily Miller' LIMIT 1) AS subquery
);

SELECT * FROM member;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| Member_ID | Name | Contact_Info | Join_Date |
|---|---|---|---|
| 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 |
| 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 |

| | | | | |
|---|---|---|---|---|
| 43 | 21:27:43 | DELETE FROM member WHERE Member_ID IN ( SELECT Member_ID FROM (SELECT Member_I... | 1 row(s) affected | 0.015 sec |
| 44 | 21:27:50 | SELECT * FROM member LIMIT 0, 1000 | 19 row(s) returned | 0.000 sec / 0.000 sec |

**10 .How do you add the following material to 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 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 Name = 'E-Books' ),
 (SELECT Genre_ID FROM Genre WHERE Name = 'Mystery &
Thriller'));
SELECT * FROM material;
```


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

| Material_ID | Title | Publication_Date | Catalog_ID | Genre_ID |
|---|---|---|---|---|
| 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 Night's Dream | 1596-01-01 | 10 | 7 |
| 31 | The Tricky Book | 1888-01-01 | 10 | 7 |
| 32 | New book | 2020-08-01 | 3 | 2 |
| NULL | NULL | NULL | NULL | NULL |

45 21:31:22 INSERT INTO Material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID) VALUES (32, 'N... 1 row(s) affected 0.250 sec
46 21:31:29 SELECT * FROM material LIMIT 0, 1000 32 row(s) returned 0.000 sec / 0.000 sec

```
INSERT INTO author (Author_ID, Name, Birth_Date, Nationality)
VALUES (21, 'Lucas Luke', '1988-09-09', 'American');
SET SQL_SAFE_UPDATES = 0;
UPDATE author
```

SET Birth_Date = NULL,
   Nationality = NULL
WHERE Name = 'Lucas Luke';
SELECT * FROM author;

```
INSERT INTO author (Author_ID, Name, Birth_Date, Nationality)
VALUES (21, 'Lucas Luke', '1988-09-09', 'American');
SET SQL_SAFE_UPDATES = 0;
UPDATE author
SET Birth_Date = NULL,
    Nationality = NULL
WHERE Name = 'Lucas Luke';
SELECT * FROM author;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

| Author_ID | Name | Birth_Date | Nationality |
|---|---|---|---|
| 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 |
| 21 | Lucas Luke | NULL | NULL |
| NULL | NULL | NULL | NULL |

| | | | | |
|---|---|---|---|---|
| 47 | 21:33:10 | INSERT INTO author (Author_ID, Name, Birth_Date, Nationality) VALUES (21, 'Lucas Luke', '1980-05-... | 1 row(s) affected | 0.375 sec |
| 48 | 21:33:10 | SELECT * FROM author LIMIT 0, 1000 | 21 row(s) returned | 0.094 sec / 0.000 sec |

INSERT INTO authorship (Authorship_ID, Author_ID, Material_ID)
VALUES (35,
 (SELECT Author_ID FROM Author WHERE Name = 'Lucas Luke'
LIMIT 1),
 (SELECT Material_ID FROM material WHERE Title = 'New book'
AND Publication_Date = '2020-08-01'));
SELECT * FROM authorship;

```
INSERT INTO authorship (Authorship_ID, Author_ID, Material_ID)
VALUES (35,
 (SELECT Author_ID FROM Author WHERE Name = 'Lucas Luke'
LIMIT 1),
 (SELECT Material_ID FROM material WHERE Title = 'New book'
AND Publication_Date = '2020-08-01'));
SELECT * FROM authorship;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

| Authorship_ID | Author_ID | Material_ID |
|---|---|---|
| 27 | 6 | 26 |
| 28 | 7 | 27 |
| 29 | 8 | 28 |
| 30 | 19 | 28 |
| 31 | 9 | 29 |
| 32 | 10 | 30 |
| 33 | 8 | 30 |
| 34 | 2 | 29 |
| 35 | 21 | 32 |
| NULL | NULL | NULL |

| | | | | |
|---|---|---|---|---|
| 49 | 21:34:55 | INSERT INTO authorship (Authorship_ID, Author_ID, Material_ID) VALUES (35, (SELECT Author_I... | 1 row(s) affected | 0.375 sec |
| 50 | 21:34:56 | SELECT * FROM authorship LIMIT 0, 1000 | 35 row(s) returned | 0.109 sec / 0.000 sec |

## 5. Design of Extended Features

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

This is an SQL query that can be implemented in the library management system. This can be used by the staff to get alerts about the overdue materials daily. This helps the staff to look at the overdue books. So the staff can alert the members to return the books. The staff can also see the title of the book and make an alert to the member.

```
SELECT ml.Title, bw.Borrow_Date, bw.Due_Date, bw.Member_ID, mem.Name AS member_name
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
JOIN member mem ON bw.Member_ID = mem.Member_ID
WHERE bw.Due_Date < CURDATE() AND bw.Return_Date IS NULL;
```

```
SELECT ml.Title, bw.Borrow_Date, bw.Due_Date, bw.Member_ID, mem.Name AS member_name
FROM borrow bw
JOIN material ml ON bw.Material_ID = ml.Material_ID
JOIN member mem ON bw.Member_ID = mem.Member_ID
WHERE bw.Due_Date < CURDATE() AND bw.Return_Date IS NULL;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| Title | Borrow_Date | Due_Date | Member_ID | member_name |
|---|---|---|---|---|
| Frankenstein | 2021-11-29 | 2021-12-20 | 1 | Alice Johnson |
| The Catcher in the Rye | 2022-12-28 | 2023-01-18 | 9 | Isla Thomas |
| To Kill a Mockingbird | 2023-01-23 | 2023-02-13 | 1 | Alice Johnson |
| The Hobbit | 2023-03-01 | 2023-03-22 | 11 | Kate Anderson |
| The Shining | 2023-03-10 | 2023-03-31 | 12 | Luke Jackson |
| Pride and Prejudice | 2023-03-15 | 2023-04-05 | 13 | Mia White |
| The Great Gatsby | 2023-03-25 | 2023-04-15 | 17 | Quinn Hall |
| Moby Dick | 2023-03-30 | 2023-04-20 | 8 | Harry Garcia |
| Crime and Punishment | 2023-03-26 | 2023-04-16 | 9 | Isla Thomas |
| The Hitchhiker's Guide to the Galaxy | 2023-03-28 | 2023-04-18 | 20 | Tiffany Allen |

63 22:10:05 SELECT ml.Title, bw.Borrow_Date, bw.Due_Date, bw.Member_ID, mem.Name AS member_name FR... 10 row(s) returned 0.000 sec / 0.000 sec

**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.**

Firstly, we would need to do a database schema modification for the member table to track the count of overdue of member and the status of the membership. We would follow these statements.

-- This is to add a column to track the overdue occurrences and membership status.

```
ALTER TABLE member
ADD COLUMN count_of_overdue INT DEFAULT 0,
ADD COLUMN active_status BOOLEAN DEFAULT TRUE;
```

The next step is to create a mechanism to increase the overdue count by 1 for each time the material is not returned on the due date. The executing statement follows below.

```
--  Here we can update the overdue for the member by 1.
UPDATE member
SET overdue_count = overdue_count + 1
WHERE Member_ID IN (
   SELECT Member_ID
   FROM borrow
```

```
   WHERE Due_Date < CURDATE() AND Return_Date IS NULL
);
```

**Deactivation of Memberships**

Here we will create one more trigger that deactivates memberships when the overdue count reaches 3 or more the membership will be deactivated.

```
-- This updates the member's overdue count and deactivates the membership.This is run on ---
-- daily basis.
UPDATE member
SET is_active = FALSE
WHERE overdue_count >= 3 AND is_active = TRUE;
```

**Reactivation of Membership**

In this, we will reactivate the member's membership after the due amount is paid by the member. This would trigger after some payment is done on the system so the member's ID is reactivated.

```
-- This is done after the payment for the over due is completed by the member.
UPDATE member
SET is_active = TRUE, overdue_count = 0
WHERE Member_ID = member;
```

# Conclusion

This database management system is used to manage and use massive datasets. This project's data relates to data from public libraries. By continuously tracking the publications, the DBMS helps to efficiently arrange the library so that staff members can oversee it and ensure that no publications are lost. It is also capable of monitoring member information on book loans from the library.

Overall, this project offers an excellent opportunity to gain knowledge about database management systems, including the creation of associations between data and how to access it. Created and inserted data, then designed queries to run the requirements based on the assignment.

**References**

[1]
Draw.io, “Flowchart Maker & Online Diagram Software,” *app.diagrams.net*, 2024. https://app.diagrams.net/

[2]
MySQL, “MySQL :: MySQL Workbench,” *Mysql.com*, 2019. https://www.mysql.com/products/workbench/

[3]
“Blackboard Learn,” *mymasonportal.gmu.edu*. https://mymasonportal.gmu.edu/ultra/courses/_512236_1/cl/outline (accessed Apr. 16, 2024).