2024 Spring CS504 Principles of Data Management and Mining Project Report

Library Management System

By Sai Sravya Chandavolu
G01454654

INTRODUCTION

Libraries are essential in educational environments because they give users access to a vast array of information, facilitate learning and research, and foster critical thinking and literacy. Books, journals, periodicals, newspapers, multimedia content, and digital resources are just a few of the varied educational resources that are available for use in libraries. These tools facilitate learning activities in a variety of areas and disciplines, both formally and informally.
Faculty, researchers, and students can get help and advice with their research from libraries. In addition to assisting users with navigating the library's resources, librarians can also help users execute efficient literary searches and assess the reliability and quality of information sources.
To help students locate, assess, and utilize information efficiently, libraries host information literacy instruction sessions. Topics include search tactics, citation guidelines, plagiarism and copyright, and digital literacy techniques are covered in these seminars. Students can conduct research, study in peace, and work together on projects in libraries' computer labs and cooperative work rooms. These areas provide a favourable setting for concentrated study and academic activity.
Overall, libraries are essential to educational settings because they give people access to materials, facilitate learning and research, support information literacy. They are vital collaborators in the learning process and have a big impact on both students' and teachers' academic progress and well-being.

SCOPE OF THE PROJECT

1. Define the scope of the project and identify the entities and their relationships.
A library management system's functions include organizing the library's resources, keeping track of everything that is borrowed, and managing the resources themselves. This project aims to design a database schema for the operation of a public library.
By using this system, we can keep track of all the information on the library's staff, book collection, and due dates.
Overseeing duties such as checking out and returning library materials.
Handling overdue books and late fees to ensure that library materials are returned on time.
The ultimate goal of this project is to give library employees a faster way to manage the resources and services offered by the library.

REQUIREMENTS

Software Requirements:
The necessary tools to finish this project is draw.io for the entity relationship diagram and MySQl for coding.

Hardware Requirements:
We do require a system with an internet connection in order to finish the project and fulfil the software requirements.

Project Requirements:

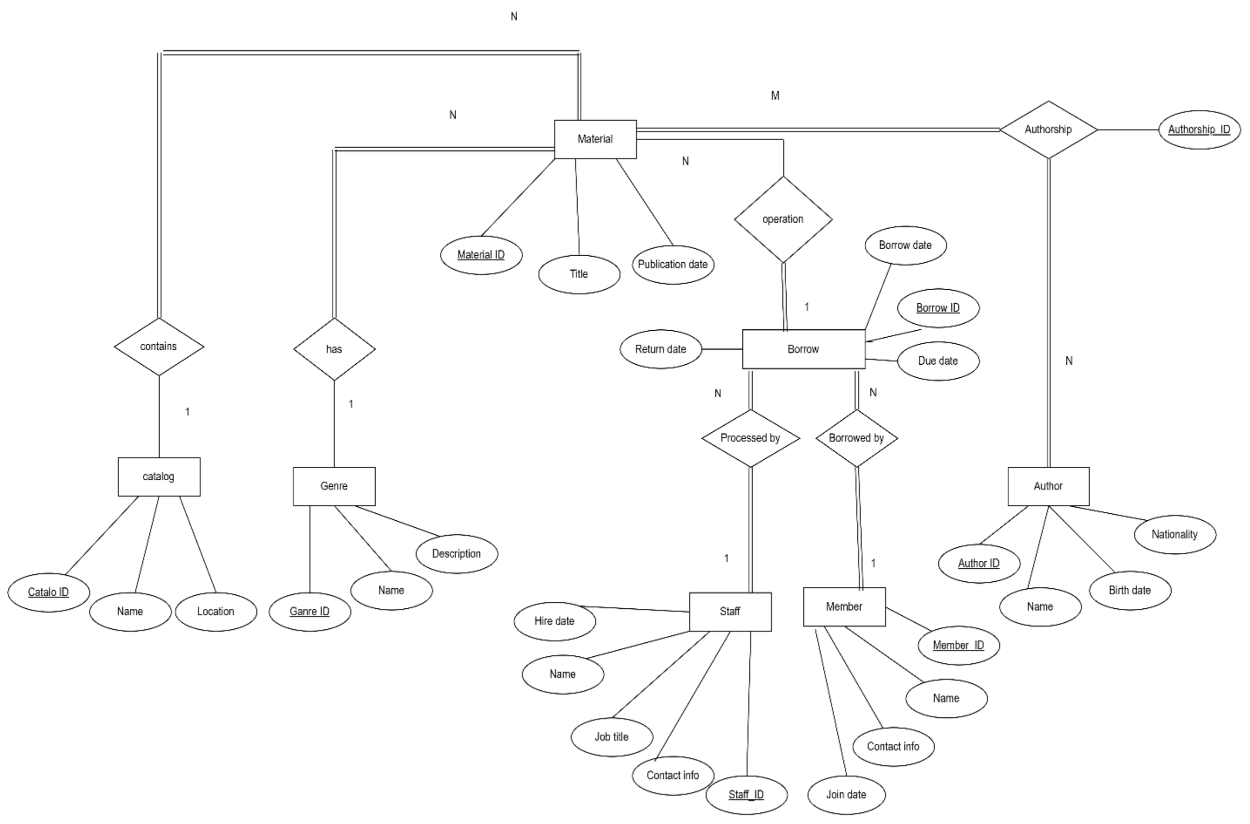
• Materials Management: The system should store and maintain information about all library materials, such as books, magazines, e-books, and audiobooks, including their titles, authors, publication dates, and genres.

• Membership Management: The system should store and manage information about library members, including their names, contact information, membership numbers, and borrowing history.

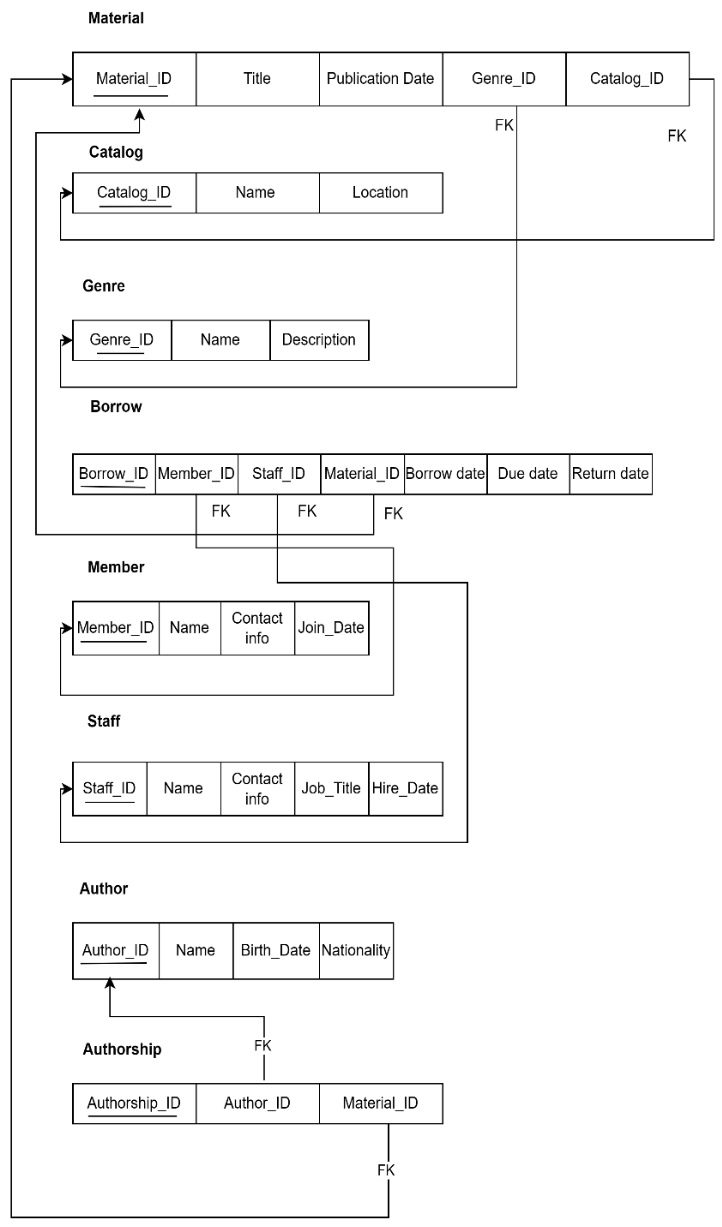
• Borrowing: The system should facilitate the borrowing process, allowing members to check out items, and provide library staff with the necessary information to manage the circulation of library materials. Once material is checked out, a librarian should record it borrow date, anticipated due date. And once the material is returned, its return date should be updated.

• Reporting and Analytics: The system should generate reports on library usage, popular materials, and other relevant statistics, enabling the library staff to make data-driven decisions about resource acquisition and management.

ENTITY RELATIONSHIP DIAGRAM / RELATIONAL SCHEMA



RELATIONAL SCHEMA:



ENTITIES AND RELATIONSHIPS:

For the Library Management System, we have the following entities.
1.Materials 2. Author 3. Staff 4. Authorship 5. Catalog 6. Genre 7. Borrow 8. Member

Relationships are as follows:
1.Contains 2. has 3. Operation 4. By 5 Derives 6. Consists

Entities:
1. Material: This refers to resources with many qualities, including Title, PublicationDate, and others, such as books, periodicals, e-books, and audio books.
2. Catalog: Information on the locations of the materials within the library is contained in this entity. The members will find it simpler to find a book they require as a result.
3. Genre: The genre of the book in the library is contained in this item. It is also possible to arrange a list of accessible content according to genre.
4. Borrow: This operation is performed to borrow books from library. This includes primary key which is Borrow_ID.
5. Member: It consists of the members who are the customers of the library.
6. Staff: This table includes all of the library's workers. They take care of the library's upkeep and guarantee that new books and borrowed materials are consistently received. The Staff_ID key is the main key.
7. Author: All of the resources in the content are written or published by the authors, who are identified by various properties such as Author_ID, Author_Name, and Publication_Date.

Relationships:
1. Contains – This relationship is between the Material and the catalog. The relationship is Many to one. Which means material can have n number of catalogues.
2. has - This relationship is between the Material and the genre. The relationship is Many to one. Which means material can have multiple genres.
3. Operation – This relationship is between the Material and Borrow. The relationship is one to many. This means that one material can be borrowed many members including staff in the library.
4. By – This relationship is between Borrow, staff and member. The relationship is many to one. This means that one staff member can borrow multiple materials.
5. Derives - This relationship is between Material and Authorship. The relationship is one to many.
6. Consists - This relationship is between Authorship and Author. The relationship is many to one.
7. Authorship: This is identified by a property which is Authorship_ID. This has relationship with author.

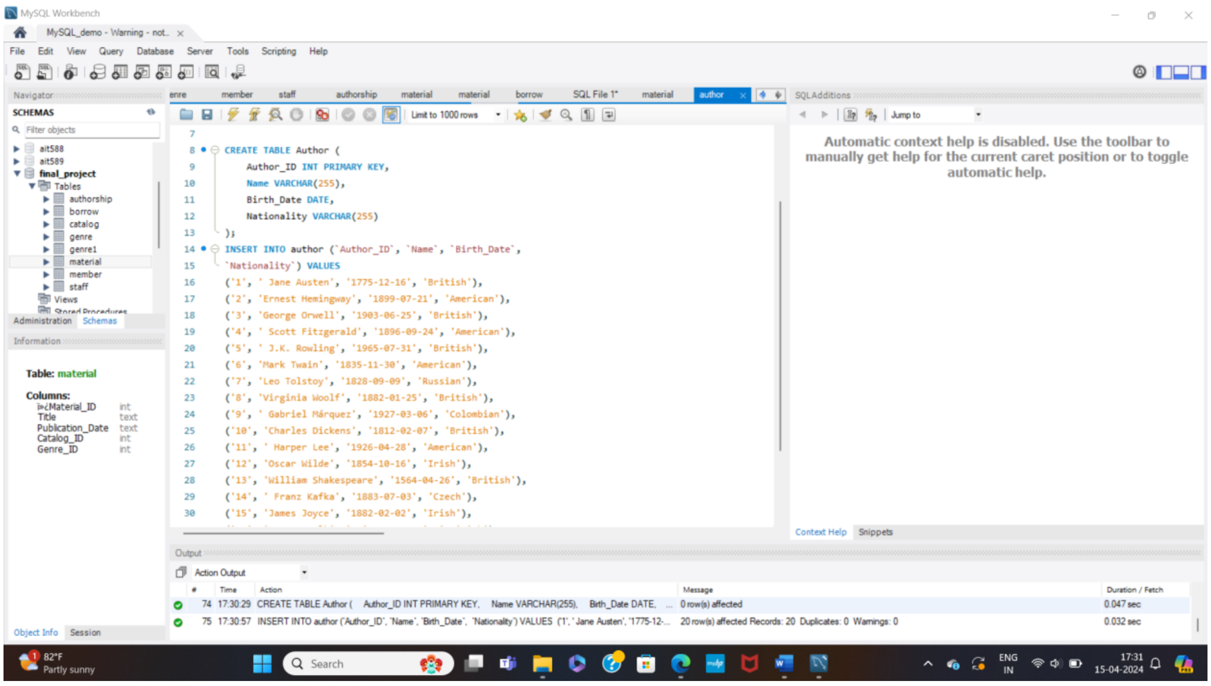
DATABASE IMPLEMENTATION:

MySQL WorkBench is used for writing the SQL queries and database connection.

Steps for creating the table and inserting the values:

Author table:

```
CREATE TABLE Author (
   Author_ID INT PRIMARY KEY,
   Name VARCHAR(255),
   Birth_Date DATE,
   Nationality VARCHAR(255)
);

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');
```



```sql
CREATE TABLE Author (
    Author_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Birth_Date DATE,
    Nationality VARCHAR(255)
);
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'),
```

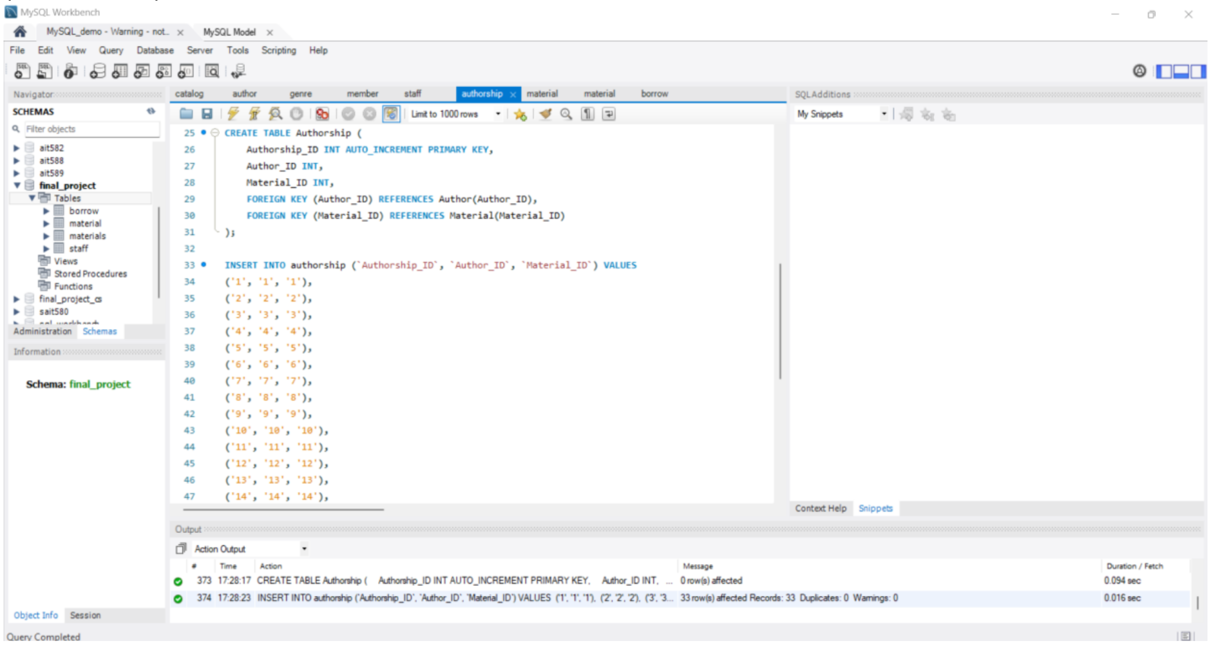
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#   Time      Action                                                                                   Message                                                        Duration / Fetch
74  17:30:29  CREATE TABLE Author (   Author_ID INT PRIMARY KEY,   Name VARCHAR(255),   Birth_Date DATE, ...   0 row(s) affected                                              0.047 sec
75  17:30:57  INSERT INTO author (`Author_ID`, `Name`, `Birth_Date`, `Nationality`) VALUES ('1', ' Jane Austen', '1775-12-...   20 row(s) affected Records: 20  Duplicates: 0  Warnings: 0   0.032 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)
);
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', '5', '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');



```
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)
);

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'),
```

```
Output
Action Output
    #    Time      Action                                                                                          Message                                                      Duration / Fetch
✓   373  17:28:17  CREATE TABLE Authorship (    Authorship_ID INT AUTO_INCREMENT PRIMARY KEY,    Author_ID INT,  ...  0 row(s) affected                                          0.094 sec
✓   374  17:28:23  INSERT INTO authorship ('Authorship_ID', 'Author_ID', 'Material_ID') VALUES ('1', '1', '1'), ('2', '2', '2'), ('3', '3...  33 row(s) affected Records: 33  Duplicates: 0  Warnings: 0  0.016 sec
```

Borrow table:

CREATE TABLE Borrow (
    Borrow_ID INT PRIMARY KEY,
    Member_ID INT,
    Material_ID INT,
    Catalog_ID INT,
    Borrow_Date DATE,
    Due_Date DATE,
    Return_Date DATE,
    FOREIGN KEY (Member_ID) REFERENCES Member(Member_ID),
    FOREIGN KEY (Material_ID) REFERENCES Material(Material_ID)
);
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);
```

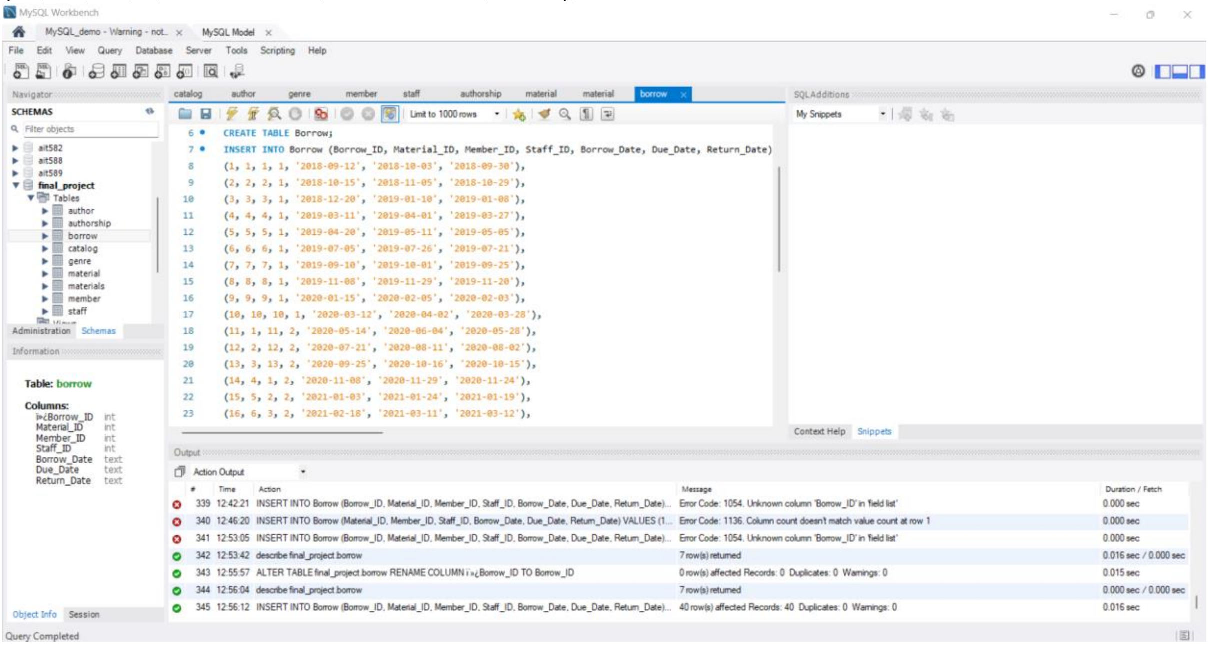


Table: borrow

Columns:

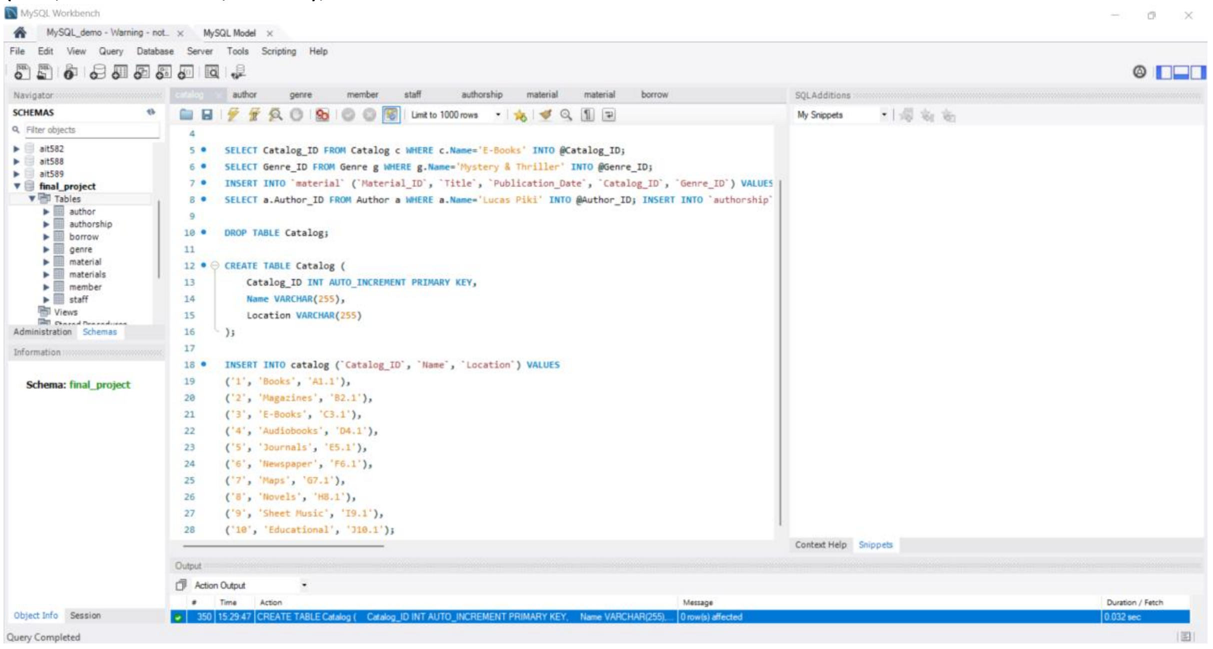
```
Borrow_ID    int
Material_ID  int
Member_ID    int
Staff_ID     int
Borrow_Date  text
Due_Date     text
Return_Date  text
```

```sql
CREATE TABLE Borrow;
INSERT INTO Borrow (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'),
(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'),
```

```
#    Time      Action                                                                                        Message                                                        Duration / Fetch
339  12:42:21  INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)...  Error Code: 1054. Unknown column 'Borrow_ID' in 'field list'   0.000 sec
340  12:46:20  INSERT INTO Borrow (Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date) VALUES (1...     Error Code: 1136. Column count doesn't match value count at row 1   0.000 sec
341  12:53:05  INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)...  Error Code: 1054. Unknown column 'Borrow_ID' in 'field list'   0.000 sec
342  12:53:42  describe final_project.borrow                                                                 7 row(s) returned                                              0.016 sec / 0.000 sec
343  12:55:57  ALTER TABLE final_project.borrow RENAME COLUMN ï»¿Borrow_ID TO Borrow_ID                          0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0       0.015 sec
344  12:56:04  describe final_project.borrow                                                                 7 row(s) returned                                              0.000 sec / 0.000 sec
345  12:56:12  INSERT INTO Borrow (Borrow_ID, Material_ID, Member_ID, Staff_ID, Borrow_Date, Due_Date, Return_Date)...  40 row(s) affected Records: 40  Duplicates: 0  Warnings: 0     0.016 sec
```

Catalog:

```
CREATE TABLE Catalog (
    Catalog_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Location VARCHAR(255)
);

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 Catalog_ID FROM Catalog c WHERE c.Name='E-Books' INTO @Catalog_ID;
SELECT Genre_ID FROM Genre g WHERE g.Name='Mystery & Thriller' INTO @Genre_ID;
INSERT INTO `material` (`Material_ID`, `Title`, `Publication_Date`, `Catalog_ID`, `Genre_ID`) VALUES
SELECT a.Author_ID FROM Author a WHERE a.Name='Lucas Piki' INTO @Author_ID; INSERT INTO `authorship`

DROP TABLE Catalog;

CREATE TABLE Catalog (
    Catalog_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Location VARCHAR(255)
);

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');
```

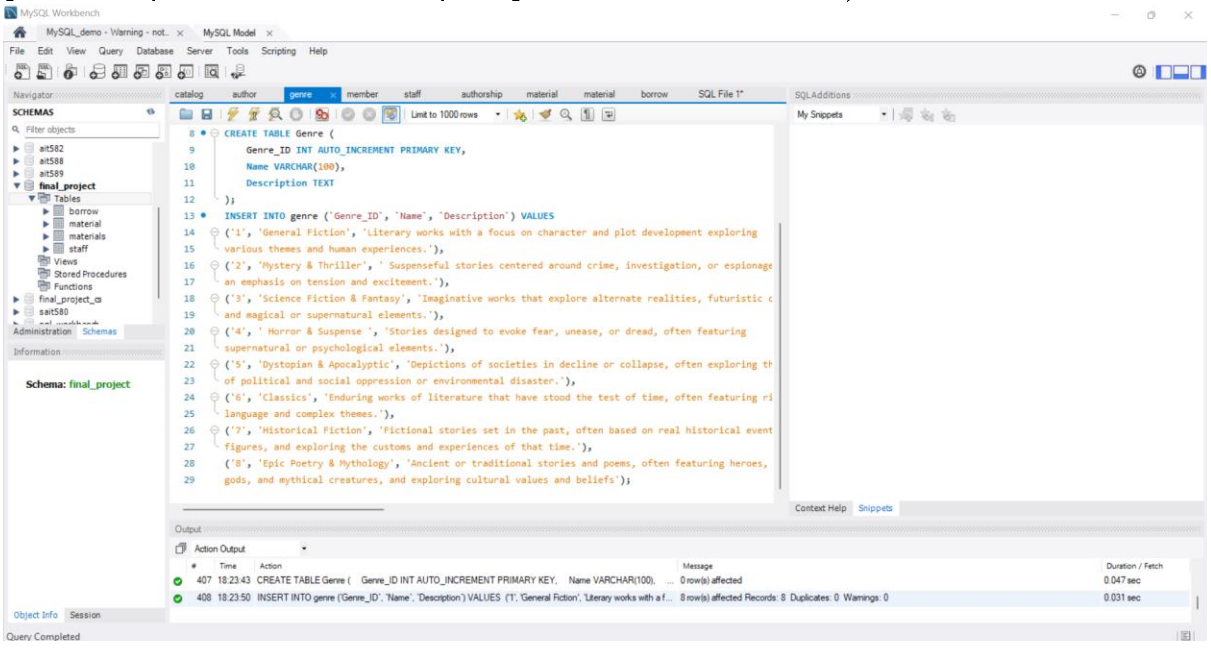
```
#    Time      Action                                                                           Message              Duration / Fetch
350  15:29:47  CREATE TABLE Catalog (    Catalog_ID INT AUTO_INCREMENT PRIMARY KEY,    Name VARCHAR(255)...  0 row(s) affected    0.032 sec
```

Query Completed

Genre table:

```
CREATE TABLE Genre (
    Genre_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Description TEXT
);
INSERT INTO genre (`Genre_ID`, `Name`, `Description`) VALUES
('1', 'General Fiction', 'Literary works with a focus on character and plot development exploring
various themes and human experiences.'),
```

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

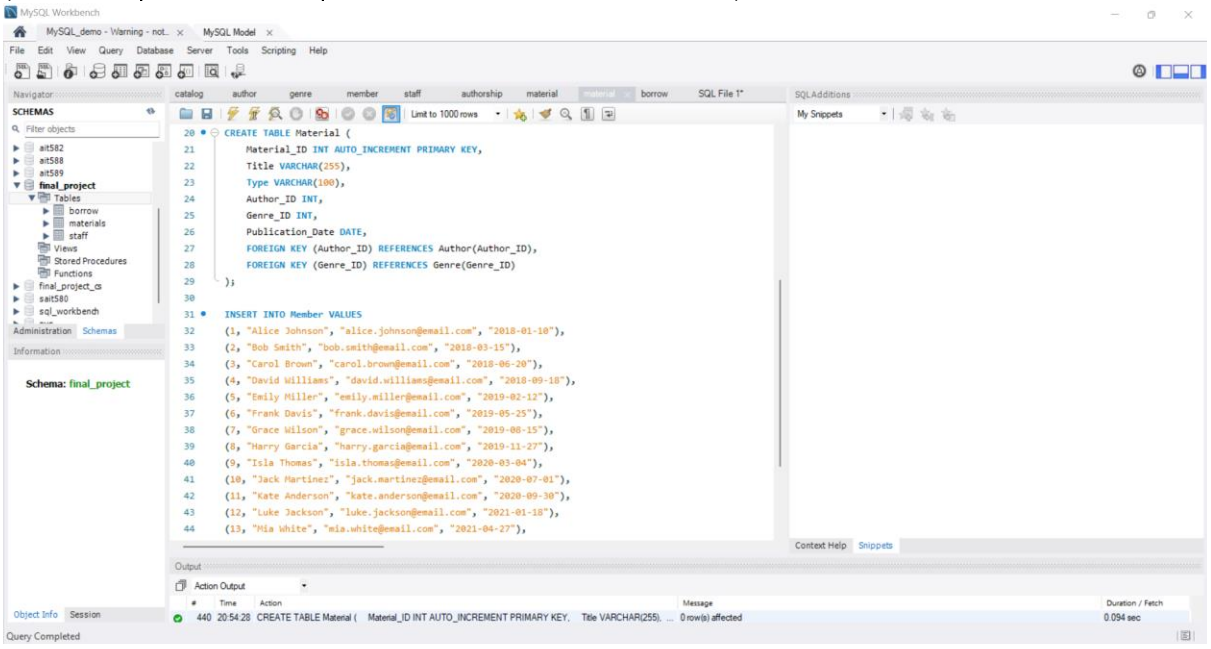

```
CREATE TABLE Genre (
    Genre_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Description TEXT
);
INSERT INTO genre (`Genre_ID`, `Name`, `Description`) VALUES
('1', 'General Fiction', 'Literary works with a focus on character and plot development exploring
various themes and human experiences.'),
('2', 'Mystery & Thriller', ' Suspenseful stories centered around crime, investigation, or espionage
an emphasis on tension and excitement.'),
('3', 'Science Fiction & Fantasy', 'Imaginative works that explore alternate realities, futuristic c
and magical or supernatural elements.'),
('4', ' Horror & Suspense ', 'Stories designed to evoke fear, unease, or dread, often featuring
supernatural or psychological elements.'),
('5', 'Dystopian & Apocalyptic', 'Depictions of societies in decline or collapse, often exploring th
of political and social oppression or environmental disaster.'),
('6', 'Classics', 'Enduring works of literature that have stood the test of time, often featuring ri
language and complex themes.'),
('7', 'Historical Fiction', 'Fictional stories set in the past, often based on real historical event
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');
```

```
#    Time      Action                                                                                  Message                                                        Duration / Fetch
407  18:23:43  CREATE TABLE Genre (   Genre_ID INT AUTO_INCREMENT PRIMARY KEY,   Name VARCHAR(100),   ...  0 row(s) affected                                              0.047 sec
408  18:23:50  INSERT INTO genre ('Genre_ID', 'Name', 'Description') VALUES ('1', 'General Fiction', 'Literary works with a f...  8 row(s) affected Records: 8  Duplicates: 0  Warnings: 0  0.031 sec
```

Material table:
CREATE TABLE Material (
  Material_ID INT AUTO_INCREMENT PRIMARY KEY,
  Title VARCHAR(255),
  Type VARCHAR(100),
  Author_ID INT,
  Genre_ID INT,
  Publication_Date DATE,
  FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID),
  FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);

INSERT INTO Member 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"),
(18, "Rachel Young", "rachel.young@email.com", "2022-06-17"),
(17, "Quinn Hall", "quinn.hall@email.com", "2022-02-28"),
(19, "Sam Walker", "sam.walker@email.com", "2022-09-25"),
(20, "Tiffany Allen", "tiffany.allen@email.com", "2022-12-10");



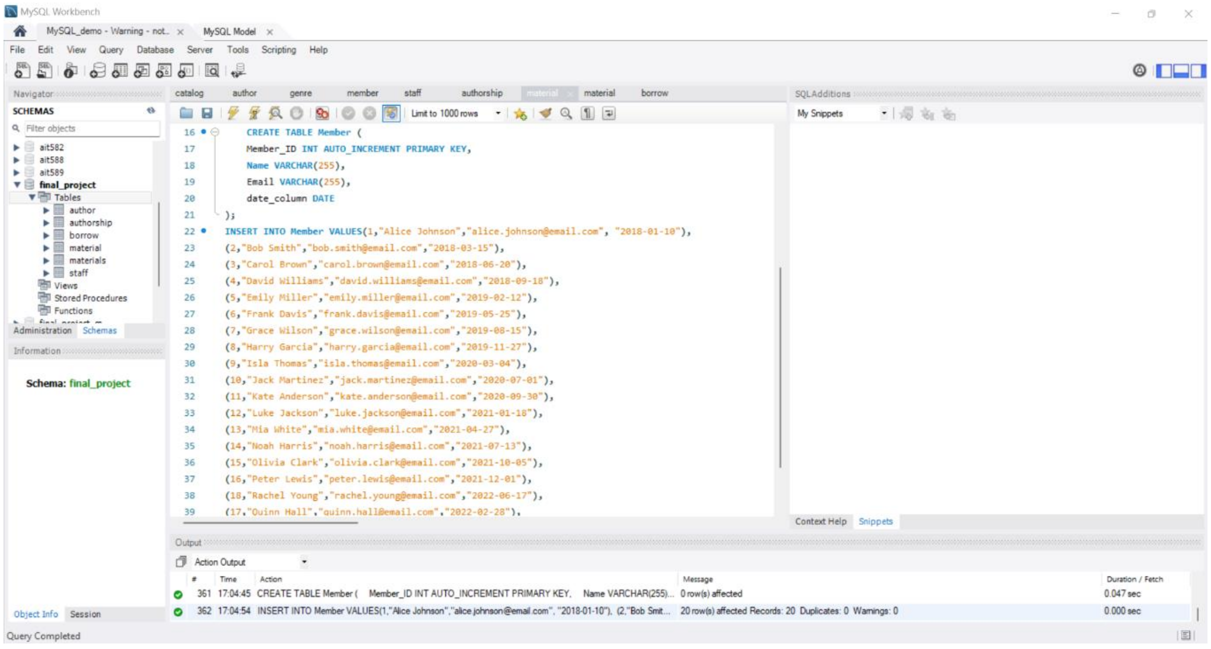
Schema: final_project

```
CREATE TABLE Material (
    Material_ID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(255),
    Type VARCHAR(100),
    Author_ID INT,
    Genre_ID INT,
    Publication_Date DATE,
    FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);

INSERT INTO Member 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"),
```

```
#    Time      Action                                                                                   Message            Duration / Fetch
440  20:54:28  CREATE TABLE Material (   Material_ID INT AUTO_INCREMENT PRIMARY KEY,   Title VARCHAR(255), ...  0 row(s) affected  0.094 sec
```

Member table:
CREATE TABLE Member (
    Member_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    date_column DATE
);
INSERT INTO Member 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"),
(18,"Rachel Young","rachel.young@email.com","2022-06-17"),
(17,"Quinn Hall","quinn.hall@email.com","2022-02-28"),
(19,"Sam Walker","sam.walker@email.com","2022-09-25"),
(20,"Tiffany Allen","tiffany.allen@email.com","2022-12-10");


```
    CREATE TABLE Member (
    Member_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255),
    date_column DATE
);
INSERT INTO Member 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"),
(18,"Rachel Young","rachel.young@email.com","2022-06-17"),
(17,"Quinn Hall","quinn.hall@email.com","2022-02-28"),
```

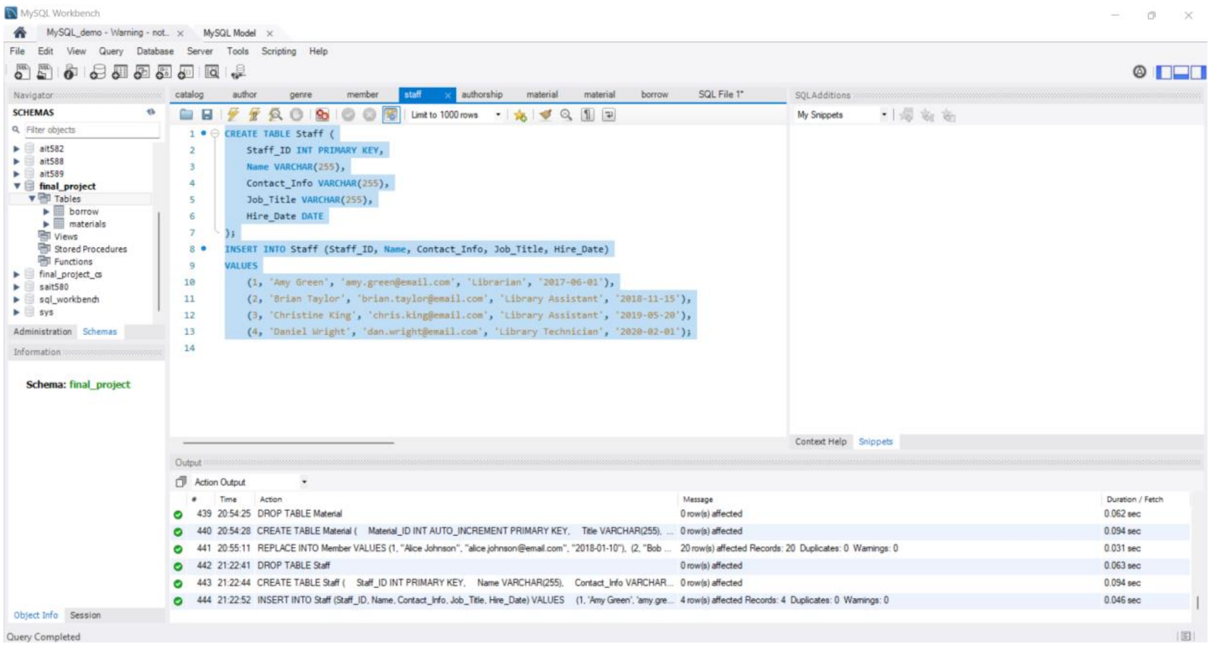
```
#    Time      Action                                                                                   Message                                                     Duration / Fetch
361  17:04:45  CREATE TABLE Member (    Member_ID INT AUTO_INCREMENT PRIMARY KEY,    Name VARCHAR(255)...  0 row(s) affected                                           0.047 sec
362  17:04:54  INSERT INTO Member VALUES(1,"Alice Johnson","alice.johnson@email.com", "2018-01-10"), (2,"Bob Smit...  20 row(s) affected Records: 20  Duplicates: 0  Warnings: 0  0.000 sec
```

Schema: final_project

Query Completed

Staff Table:

CREATE TABLE Staff (
    Staff_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Contact_Info VARCHAR(255),
    Job_Title VARCHAR(255),
    Hire_Date DATE
);
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');



```sql
CREATE TABLE Staff (
    Staff_ID INT PRIMARY KEY,
    Name VARCHAR(255),
    Contact_Info VARCHAR(255),
    Job_Title VARCHAR(255),
    Hire_Date DATE
);
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');
```

Schema: final_project

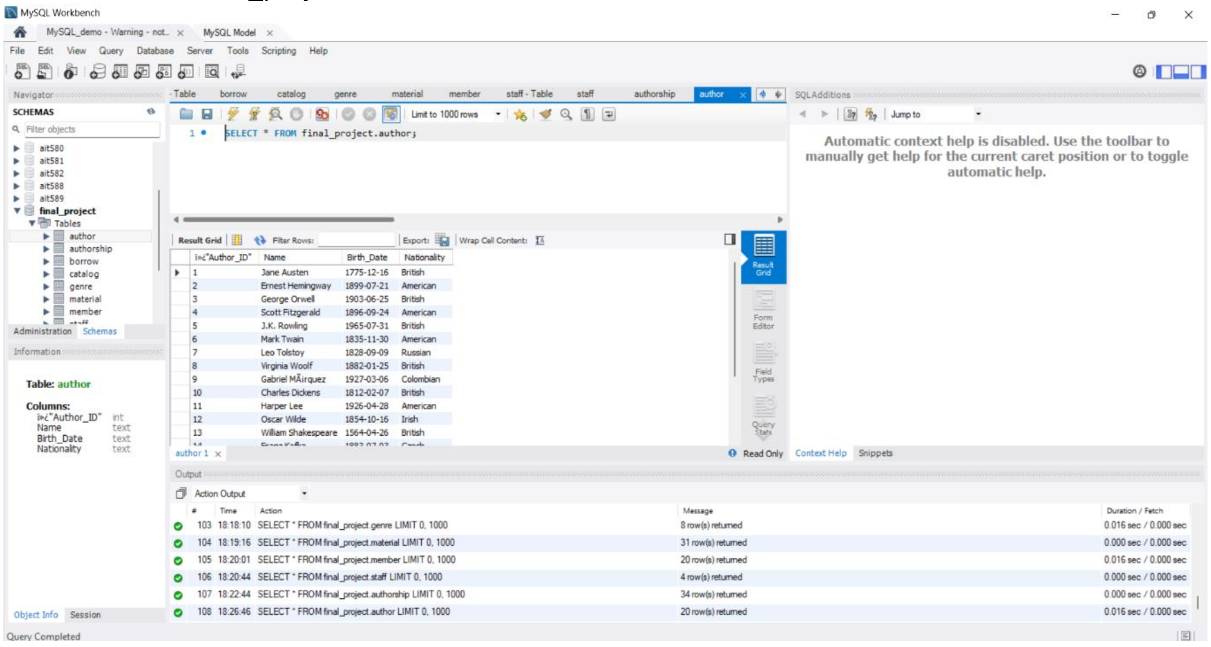
```
#    Time      Action                                                                                      Message                                                        Duration / Fetch
439  20:54:25  DROP TABLE Material                                                                         0 row(s) affected                                              0.062 sec
440  20:54:28  CREATE TABLE Material (   Material_ID INT AUTO_INCREMENT PRIMARY KEY,   Title VARCHAR(255), ...  0 row(s) affected                                         0.094 sec
441  20:55:11  REPLACE INTO Member VALUES (1, "Alice Johnson", "alice.johnson@email.com", "2018-01-10"), (2, "Bob ...  20 row(s) affected Records: 20  Duplicates: 0  Warnings: 0  0.031 sec
442  21:22:41  DROP TABLE Staff                                                                            0 row(s) affected                                              0.063 sec
443  21:22:44  CREATE TABLE Staff (   Staff_ID INT PRIMARY KEY,   Name VARCHAR(255),   Contact_Info VARCHAR...  0 row(s) affected                                         0.094 sec
444  21:22:52  INSERT INTO Staff (Staff_ID, Name, Contact_Info, Job_Title, Hire_Date) VALUES   (1, 'Amy Green', 'amy gre...  4 row(s) affected Records: 4  Duplicates: 0  Warnings: 0  0.046 sec
```

//This is how we create and insert tables manually. But the method I used to create data is by exporting files directly from the path to SQL workbench. //

VIEWING TABLES:

Author table:

SELECT * FROM final_project.author;



```sql
SELECT * FROM final_project.author;
```

ï»¿"Author_ID"	Name	Birth_Date	Nationality
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 MÃirquez	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

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

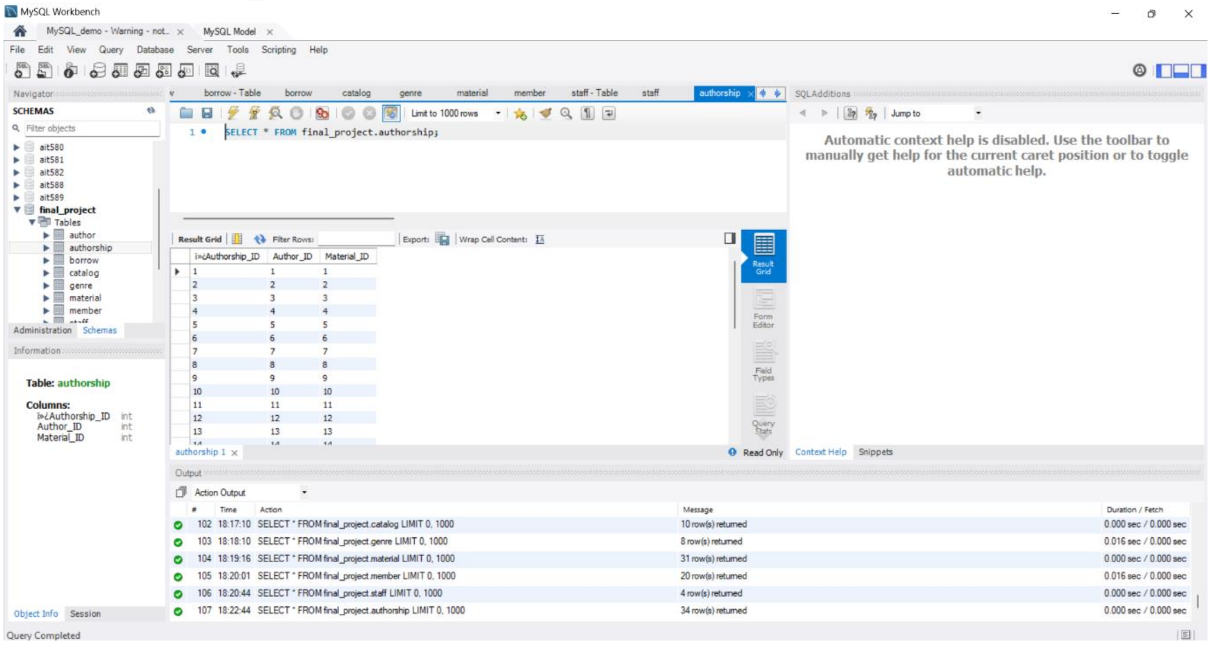
Table: author

Columns:
ï»¿"Author_ID" int
Name text
Birth_Date text
Nationality text

```
#    Time      Action                                                  Message             Duration / Fetch
103  18:18:10  SELECT * FROM final_project.genre LIMIT 0, 1000         8 row(s) returned   0.016 sec / 0.000 sec
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000      31 row(s) returned  0.000 sec / 0.000 sec
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000        20 row(s) returned  0.016 sec / 0.000 sec
106  18:20:44  SELECT * FROM final_project.staff LIMIT 0, 1000         4 row(s) returned   0.000 sec / 0.000 sec
107  18:22:44  SELECT * FROM final_project.authorship LIMIT 0, 1000    34 row(s) returned  0.000 sec / 0.000 sec
108  18:26:46  SELECT * FROM final_project.author LIMIT 0, 1000        20 row(s) returned  0.016 sec / 0.000 sec
```

Authorship table:

SELECT * FROM final_project.authorship;



```
SELECT * FROM final_project.authorship;
```

ï»¿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
11	11	11
12	12	12
13	13	13

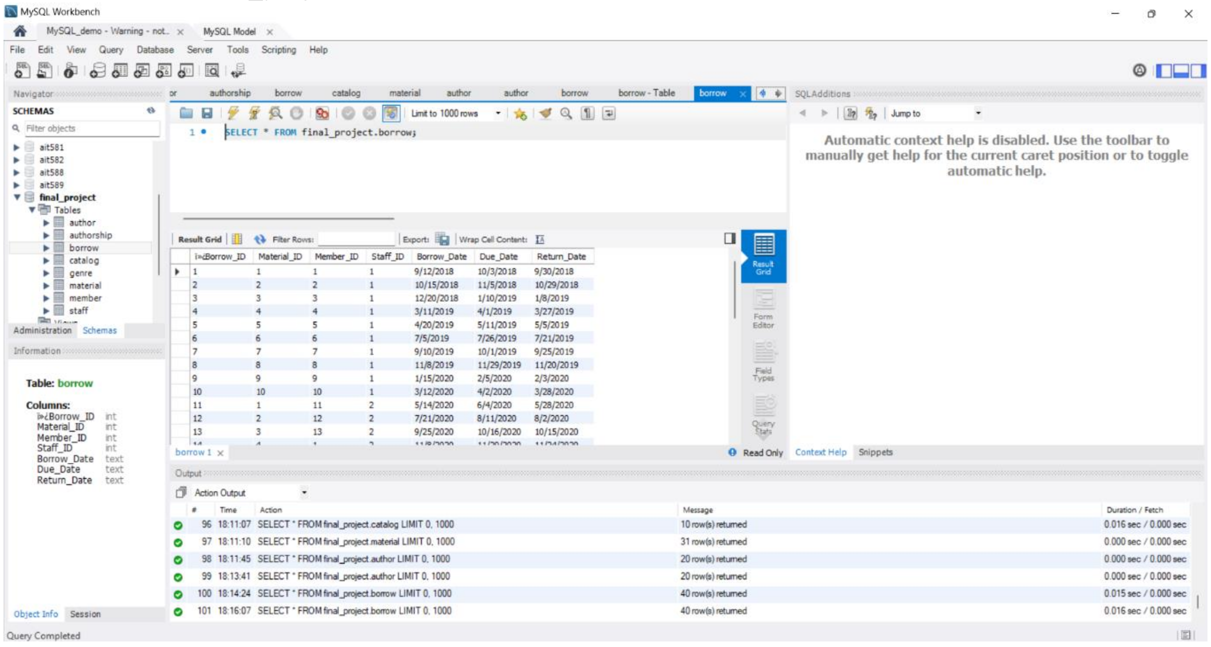
Table: authorship

Columns:
ï»¿Authorship_ID int
Author_ID int
Material_ID int

```
#    Time      Action                                              Message            Duration / Fetch
102  18:17:10  SELECT * FROM final_project.catalog LIMIT 0, 1000     10 row(s) returned  0.000 sec / 0.000 sec
103  18:18:10  SELECT * FROM final_project.genre LIMIT 0, 1000       8 row(s) returned   0.016 sec / 0.000 sec
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000    31 row(s) returned  0.000 sec / 0.000 sec
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000      20 row(s) returned  0.016 sec / 0.000 sec
106  18:20:44  SELECT * FROM final_project.staff LIMIT 0, 1000       4 row(s) returned   0.000 sec / 0.000 sec
107  18:22:44  SELECT * FROM final_project.authorship LIMIT 0, 1000  34 row(s) returned  0.000 sec / 0.000 sec
```

Borrow table:

SELECT * FROM final_project.borrow;



```
SELECT * FROM final_project.borrow;
```

ï»¿Borrow_ID	Material_ID	Member_ID	Staff_ID	Borrow_Date	Due_Date	Return_Date
1	1	1	1	9/12/2018	10/3/2018	9/30/2018
2	2	2	1	10/15/2018	11/5/2018	10/29/2018
3	3	3	1	12/20/2018	1/10/2019	1/8/2019
4	4	4	1	3/11/2019	4/1/2019	3/27/2019
5	5	5	1	4/20/2019	5/11/2019	5/5/2019
6	6	6	1	7/5/2019	7/26/2019	7/21/2019
7	7	7	1	9/10/2019	10/1/2019	9/25/2019
8	8	8	1	11/8/2019	11/29/2019	11/20/2019
9	9	9	1	1/15/2020	2/5/2020	2/3/2020
10	10	10	1	3/12/2020	4/2/2020	3/28/2020
11	1	11	2	5/14/2020	6/4/2020	5/28/2020
12	2	12	2	7/21/2020	8/11/2020	8/2/2020
13	3	13	2	9/25/2020	10/16/2020	10/15/2020

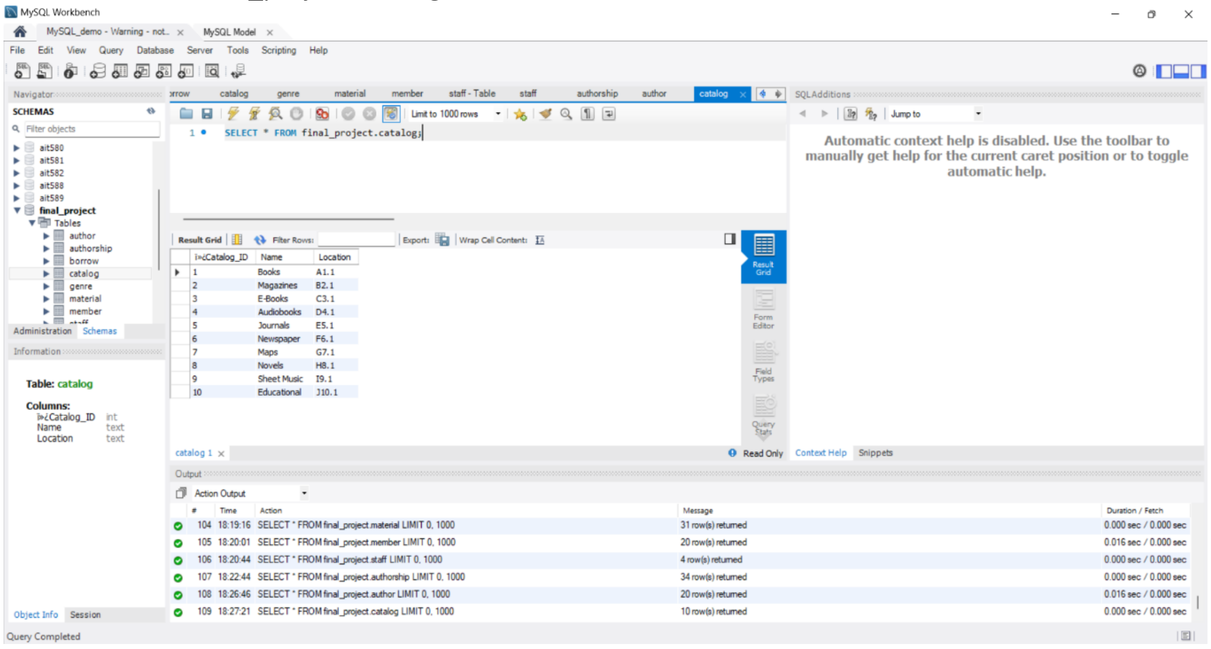
Table: borrow

Columns:
ï»¿Borrow_ID int
Material_ID int
Member_ID int
Staff_ID int
Borrow_Date text
Due_Date text
Return_Date text

```
#    Time      Action                                            Message            Duration / Fetch
96   18:11:07  SELECT * FROM final_project.catalog LIMIT 0, 1000   10 row(s) returned  0.016 sec / 0.000 sec
97   18:11:10  SELECT * FROM final_project.material LIMIT 0, 1000  31 row(s) returned  0.000 sec / 0.000 sec
98   18:11:45  SELECT * FROM final_project.author LIMIT 0, 1000    20 row(s) returned  0.000 sec / 0.000 sec
99   18:13:41  SELECT * FROM final_project.author LIMIT 0, 1000    20 row(s) returned  0.000 sec / 0.000 sec
100  18:14:24  SELECT * FROM final_project.borrow LIMIT 0, 1000    40 row(s) returned  0.015 sec / 0.000 sec
101  18:16:07  SELECT * FROM final_project.borrow LIMIT 0, 1000    40 row(s) returned  0.016 sec / 0.000 sec
```

Catalog table:

SELECT * FROM final_project.catalog;



```sql
SELECT * FROM final_project.catalog;
```

ï»¿Catalog_ID	Name	Location
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

Table: catalog

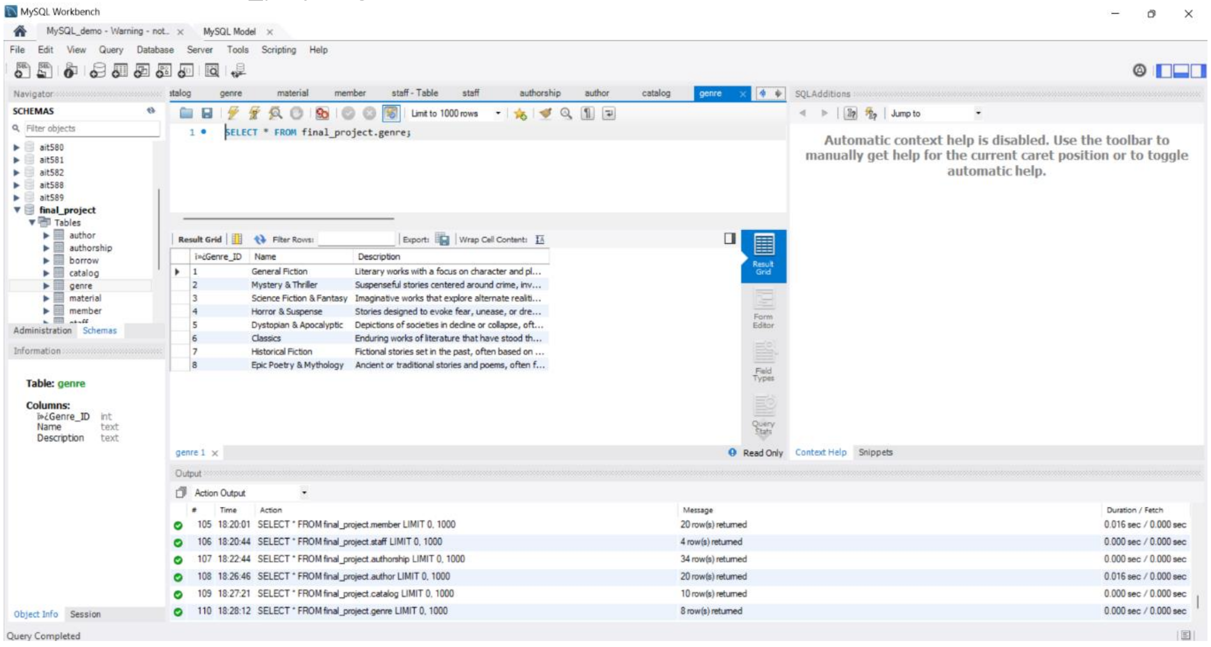
Columns:
ï»¿Catalog_ID int
Name text
Location text

```
#    Time      Action                                                   Message             Duration / Fetch
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000       31 row(s) returned  0.000 sec / 0.000 sec
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000         20 row(s) returned  0.016 sec / 0.000 sec
106  18:20:44  SELECT * FROM final_project.staff LIMIT 0, 1000          4 row(s) returned   0.000 sec / 0.000 sec
107  18:22:44  SELECT * FROM final_project.authorship LIMIT 0, 1000     34 row(s) returned  0.000 sec / 0.000 sec
108  18:26:46  SELECT * FROM final_project.author LIMIT 0, 1000         20 row(s) returned  0.016 sec / 0.000 sec
109  18:27:21  SELECT * FROM final_project.catalog LIMIT 0, 1000        10 row(s) returned  0.000 sec / 0.000 sec
```

Query Completed

Genre table:

SELECT * FROM final_project.genre;



```sql
SELECT * FROM final_project.genre;
```

ï»¿Genre_ID	Name	Description
1	General Fiction	Literary works with a focus on character and pl...
2	Mystery & Thriller	Suspenseful stories centered around crime, inv...
3	Science Fiction & Fantasy	Imaginative works that explore alternate realiti...
4	Horror & Suspense	Stories designed to evoke fear, unease, or dre...
5	Dystopian & Apocalyptic	Depictions of societies in decline or collapse, oft...
6	Classics	Enduring works of literature that have stood th...
7	Historical Fiction	Fictional stories set in the past, often based on ...
8	Epic Poetry & Mythology	Ancient or traditional stories and poems, often f...

Table: genre

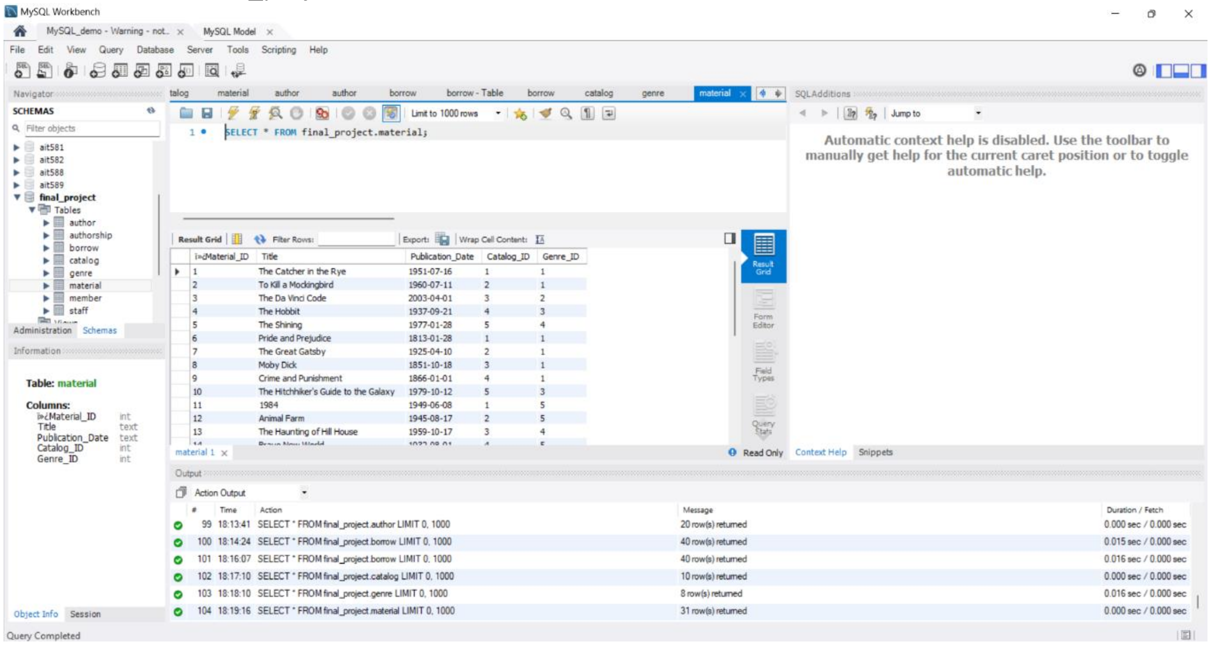
Columns:
ï»¿Genre_ID int
Name text
Description text

```
#    Time      Action                                                   Message             Duration / Fetch
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000         20 row(s) returned  0.016 sec / 0.000 sec
106  18:20:44  SELECT * FROM final_project.staff LIMIT 0, 1000          4 row(s) returned   0.000 sec / 0.000 sec
107  18:22:44  SELECT * FROM final_project.authorship LIMIT 0, 1000     34 row(s) returned  0.000 sec / 0.000 sec
108  18:26:46  SELECT * FROM final_project.author LIMIT 0, 1000         20 row(s) returned  0.016 sec / 0.000 sec
109  18:27:21  SELECT * FROM final_project.catalog LIMIT 0, 1000        10 row(s) returned  0.000 sec / 0.000 sec
110  18:28:12  SELECT * FROM final_project.genre LIMIT 0, 1000          8 row(s) returned   0.000 sec / 0.000 sec
```

Query Completed

Material table:

SELECT * FROM final_project.material;



```
SELECT * FROM final_project.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	1925-04-10	2	1
8	Moby Dick	1851-10-18	3	1
9	Crime and Punishment	1866-01-01	4	1
10	The Hitchhiker's Guide to the Galaxy	1979-10-12	5	3
11	1984	1949-06-08	1	5
12	Animal Farm	1945-08-17	2	5
13	The Haunting of Hill House	1959-10-17	3	4

Table: material

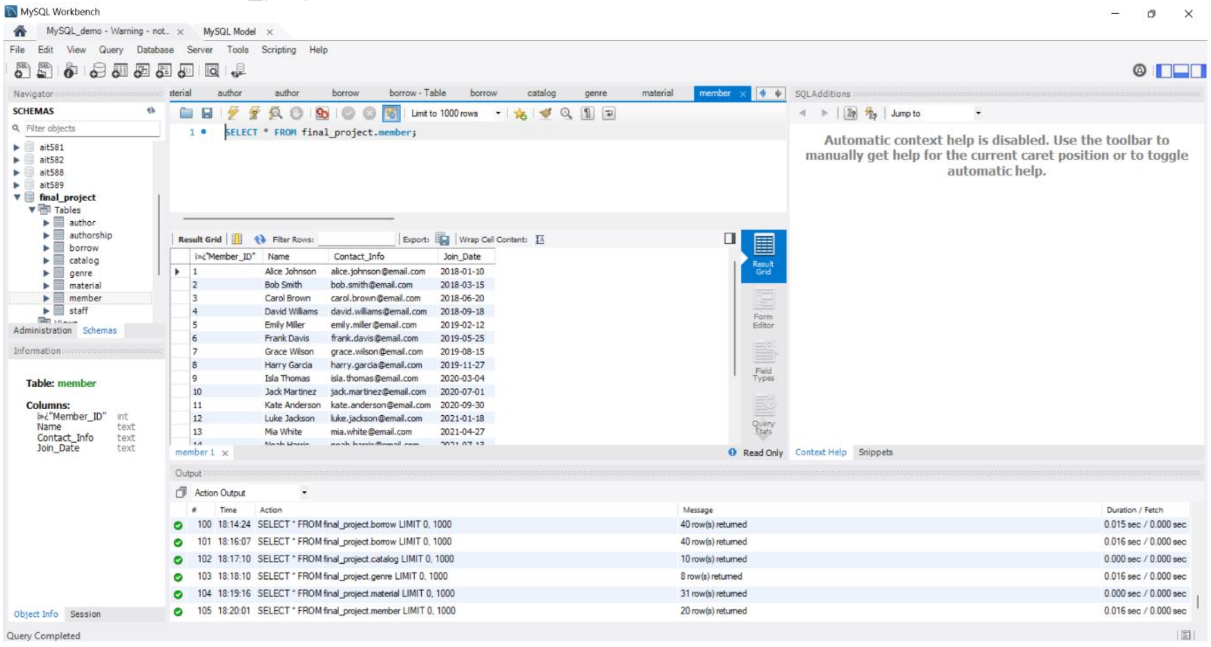
Columns:
Material_ID int
Title text
Publication_Date text
Catalog_ID int
Genre_ID int

```
#    Time      Action                                               Message              Duration / Fetch
99   18:13:41  SELECT * FROM final_project.author LIMIT 0, 1000     20 row(s) returned   0.000 sec / 0.000 sec
100  18:14:24  SELECT * FROM final_project.borrow LIMIT 0, 1000     40 row(s) returned   0.015 sec / 0.000 sec
101  18:16:07  SELECT * FROM final_project.borrow LIMIT 0, 1000     40 row(s) returned   0.016 sec / 0.000 sec
102  18:17:10  SELECT * FROM final_project.catalog LIMIT 0, 1000    10 row(s) returned   0.000 sec / 0.000 sec
103  18:18:10  SELECT * FROM final_project.genre LIMIT 0, 1000      8 row(s) returned    0.016 sec / 0.000 sec
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000   31 row(s) returned   0.000 sec / 0.000 sec
```

Query Completed

Member table:

SELECT * FROM final_project.member;



```
SELECT * FROM final_project.member;
```

"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
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

Table: member

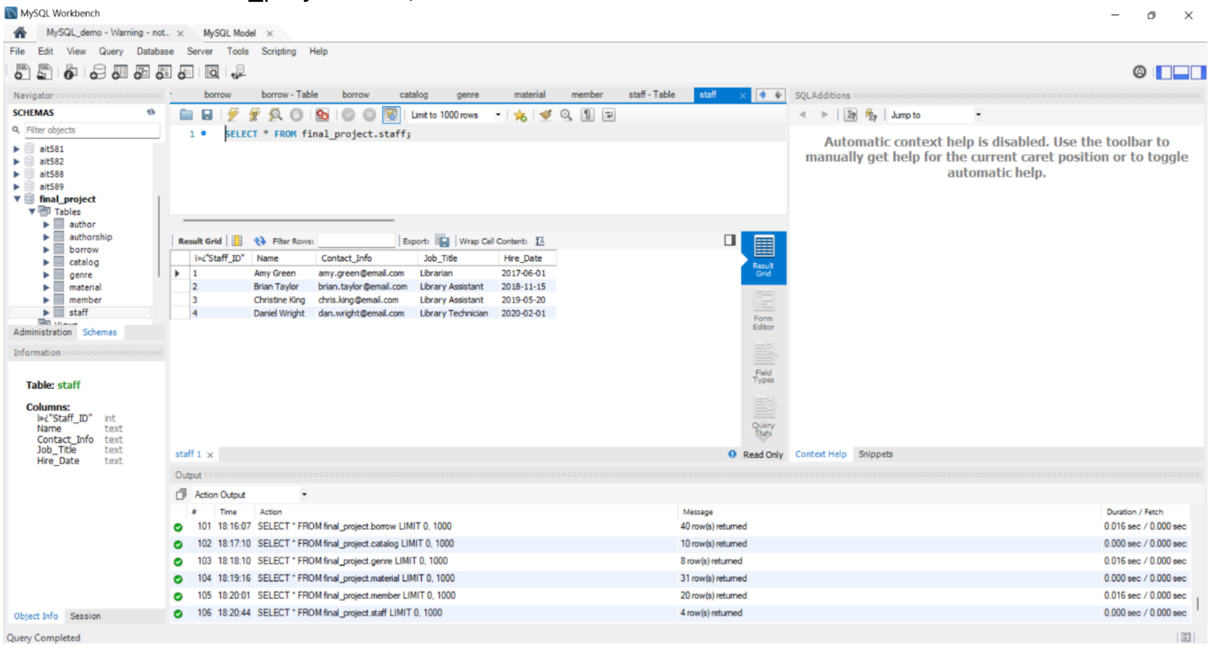
Columns:
"Member_ID" int
Name text
Contact_Info text
Join_Date text

```
#    Time      Action                                               Message              Duration / Fetch
100  18:14:24  SELECT * FROM final_project.borrow LIMIT 0, 1000     40 row(s) returned   0.015 sec / 0.000 sec
101  18:16:07  SELECT * FROM final_project.borrow LIMIT 0, 1000     40 row(s) returned   0.016 sec / 0.000 sec
102  18:17:10  SELECT * FROM final_project.catalog LIMIT 0, 1000    10 row(s) returned   0.000 sec / 0.000 sec
103  18:18:10  SELECT * FROM final_project.genre LIMIT 0, 1000      8 row(s) returned    0.016 sec / 0.000 sec
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000   31 row(s) returned   0.000 sec / 0.000 sec
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000     20 row(s) returned   0.016 sec / 0.000 sec
```

Query Completed

Staff table:

SELECT * FROM final_project.staff;



```
SELECT * FROM final_project.staff;
```

ï»¿"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

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: staff

Columns:
```
ï»¿"Staff_ID"  int
Name          text
Contact_Info  text
Job_Title     text
Hire_Date     text
```

```
#    Time      Action                                              Message             Duration / Fetch
101  18:16:07  SELECT * FROM final_project.borrow LIMIT 0, 1000    40 row(s) returned  0.016 sec / 0.000 sec
102  18:17:10  SELECT * FROM final_project.catalog LIMIT 0, 1000   10 row(s) returned  0.000 sec / 0.000 sec
103  18:18:10  SELECT * FROM final_project.genre LIMIT 0, 1000     8 row(s) returned   0.016 sec / 0.000 sec
104  18:19:16  SELECT * FROM final_project.material LIMIT 0, 1000  31 row(s) returned  0.000 sec / 0.000 sec
105  18:20:01  SELECT * FROM final_project.member LIMIT 0, 1000    20 row(s) returned  0.016 sec / 0.000 sec
106  18:20:44  SELECT * FROM final_project.staff LIMIT 0, 1000     4 row(s) returned   0.000 sec / 0.000 sec
```

Query Completed

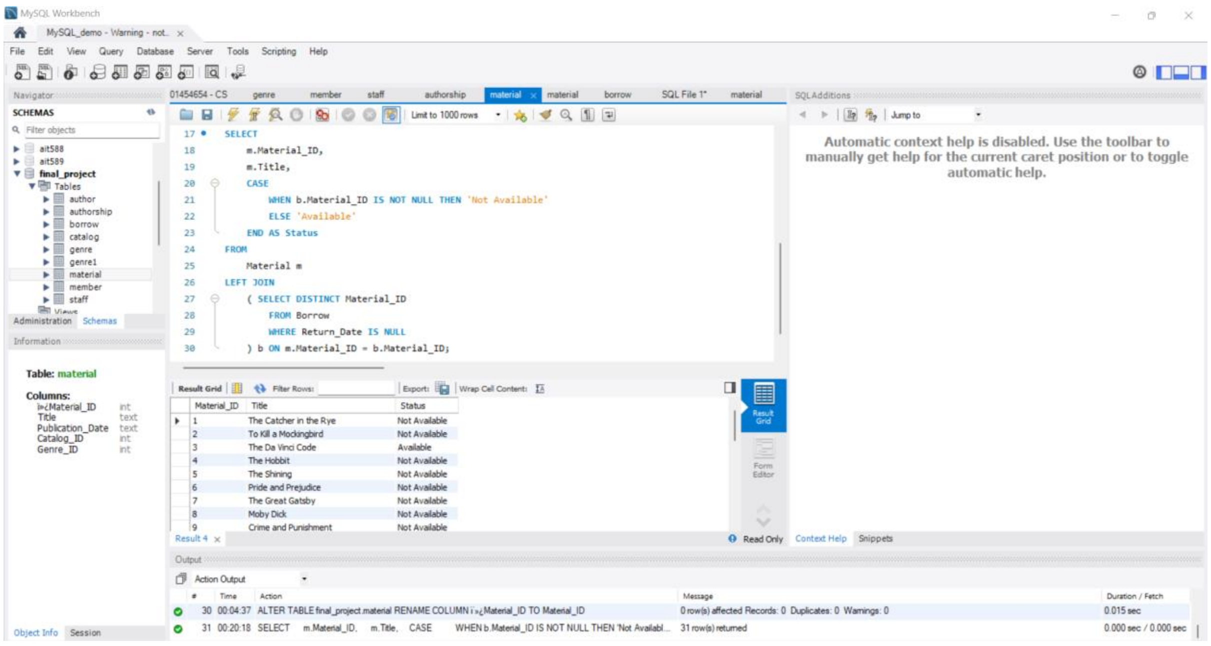
QUERIES

Query 1:

Which materials are currently available in the library?

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

```
SELECT m.Material_ID,
m.Title,
CASE
    WHEN b.material_ID IS NOT NULL THEN 'Not_available'
    ELSE 'Available'
END AS Status
FROM
Material m
LEFT JOIN
(
SELECT DISTINCT Material_ID
FROM Borrow
WHERE Return_Date IS NULL
) b ON m.Material_ID=b.Material_ID;
```



```sql
SELECT
    m.Material_ID,
    m.Title,
    CASE
        WHEN b.Material_ID IS NOT NULL THEN 'Not Available'
        ELSE 'Available'
    END AS Status
FROM
    Material m
LEFT JOIN
    ( SELECT DISTINCT Material_ID
        FROM Borrow
        WHERE Return_Date IS NULL
    ) b ON m.Material_ID = b.Material_ID;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: material

Columns:
```
ï»¿Material_ID     int
Title             text
Publication_Date  text
Catalog_ID        int
Genre_ID          int
```

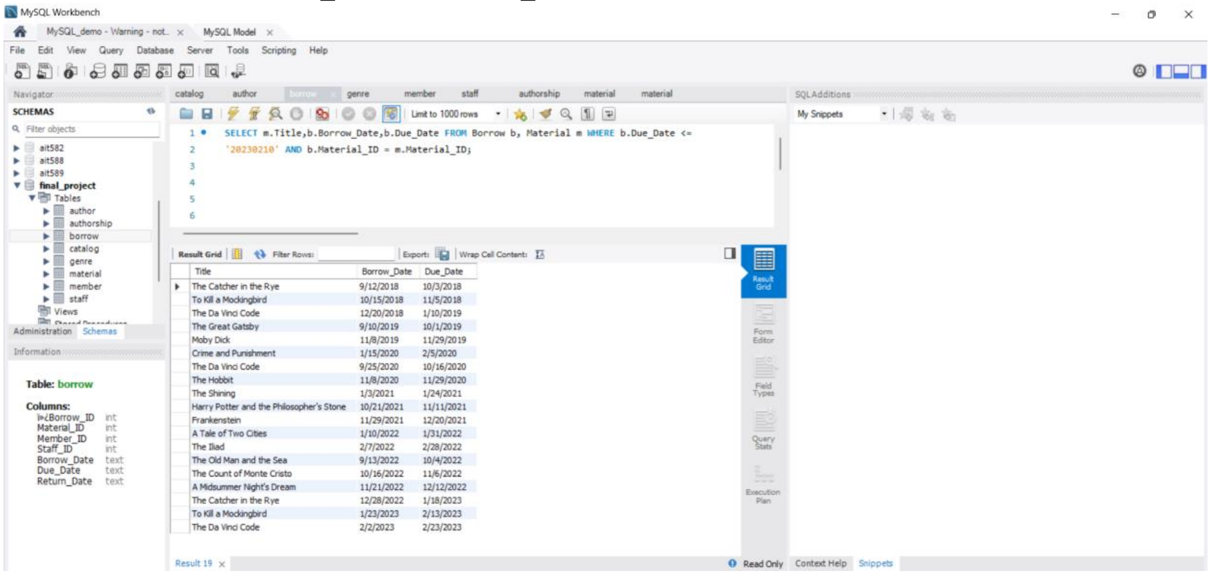
Material_ID	Title	Status
1	The Catcher in the Rye	Not Available
2	To Kill a Mockingbird	Not Available
3	The Da Vinci Code	Available
4	The Hobbit	Not Available
5	The Shining	Not Available
6	Pride and Prejudice	Not Available
7	The Great Gatsby	Not Available
8	Moby Dick	Not Available
9	Crime and Punishment	Not Available

```
#   Time      Action                                                                          Message                                                        Duration / Fetch
30  00:04:37  ALTER TABLE final_project.material RENAME COLUMN ï»¿Material_ID TO Material_ID    0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0   0.015 sec
31  00:20:18  SELECT  m.Material_ID,  m.Title,  CASE  WHEN b.Material_ID IS NOT NULL THEN 'Not Availabl...   31 row(s) returned                    0.000 sec / 0.000 sec
```

Query 2:

Which materials are currently overdue?

SELECT m.Title,b.Borrow_Date,b.Due_Date FROM Borrow b, Material m WHERE b.Due_Date <= ' 20230210' AND b.Material_ID = m.Material_ID;



```sql
SELECT m.Title,b.Borrow_Date,b.Due_Date FROM Borrow b, Material m WHERE b.Due_Date <=
'20230210' AND b.Material_ID = m.Material_ID;
```

Table: borrow

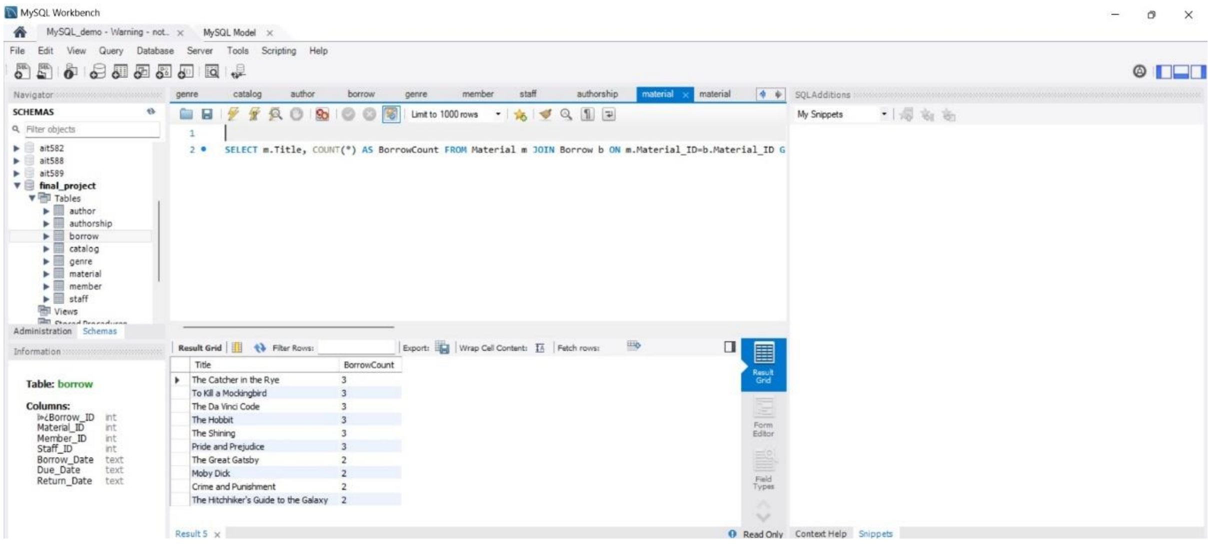
Columns:
```
ï»¿Borrow_ID   int
Material_ID   int
Member_ID     int
Staff_ID      int
Borrow_Date   text
Due_Date      text
Return_Date   text
```

Title	Borrow_Date	Due_Date
The Catcher in the Rye	9/12/2018	10/3/2018
To Kill a Mockingbird	10/15/2018	11/5/2018
The Da Vinci Code	12/20/2018	1/10/2019
The Great Gatsby	9/10/2019	10/1/2019
Moby Dick	11/8/2019	11/29/2019
Crime and Punishment	1/15/2020	2/5/2020
The Da Vinci Code	9/25/2020	10/16/2020
The Hobbit	11/8/2020	11/29/2020
The Shining	1/3/2021	1/24/2021
Harry Potter and the Philosopher's Stone	10/21/2021	11/11/2021
Frankenstein	11/29/2021	12/20/2021
A Tale of Two Cities	1/10/2022	1/31/2022
The Iliad	2/7/2022	2/28/2022
The Old Man and the Sea	9/13/2022	10/4/2022
The Count of Monte Cristo	10/16/2022	11/6/2022
A Midsummer Night's Dream	11/21/2022	12/12/2022
The Catcher in the Rye	12/28/2022	1/18/2023
To Kill a Mockingbird	1/23/2023	2/13/2023
The Da Vinci Code	2/2/2023	2/23/2023

Query 3:

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

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



```
SELECT m.Title, COUNT(*) AS BorrowCount FROM Material m JOIN Borrow b ON m.Material_ID=b.Material_ID G
```

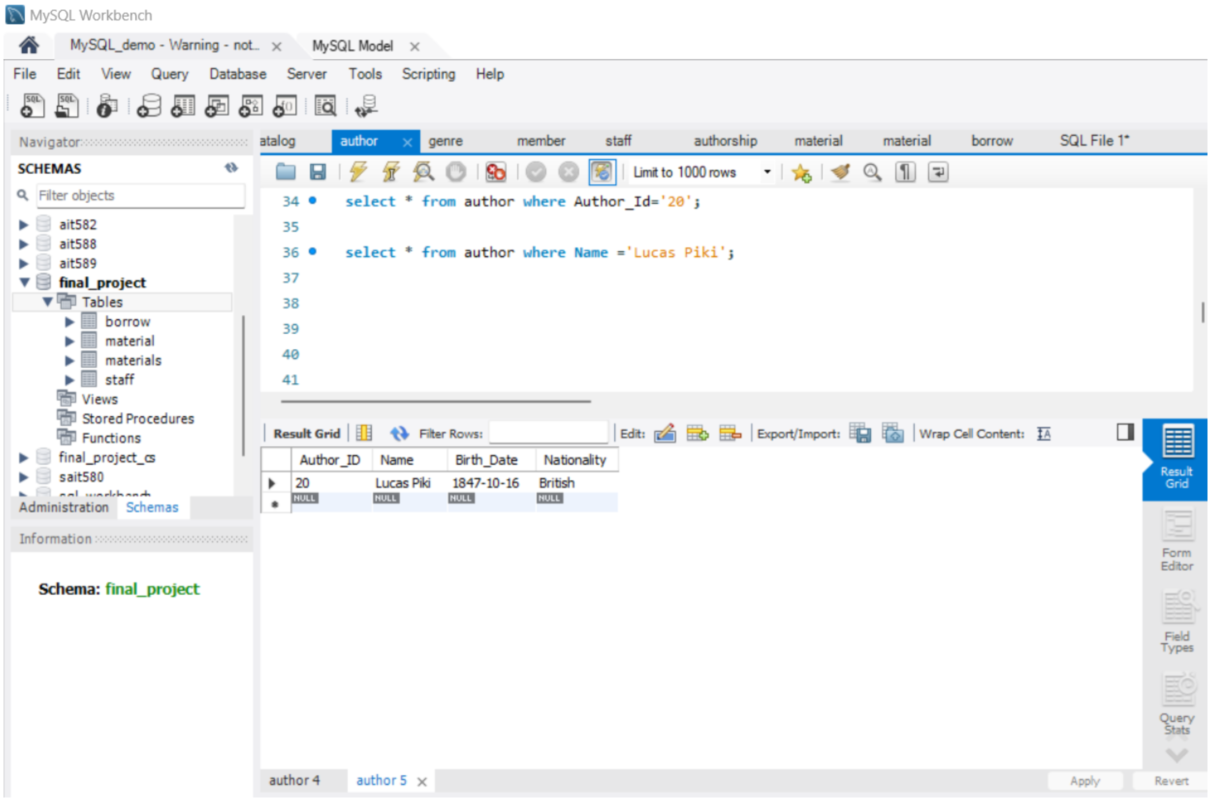
Title	BorrowCount
The Catcher in the Rye	3
To Kill a Mockingbird	3
The Da Vinci Code	3
The Hobbit	3
The Shining	3
Pride and Prejudice	3
The Great Gatsby	2
Moby Dick	2
Crime and Punishment	2
The Hitchhiker's Guide to the Galaxy	2

Query 4:

How many books has the author Lucas Piki written?

select * from author where Author_Id='20';

select * from author where Name ='Lucas Piki';



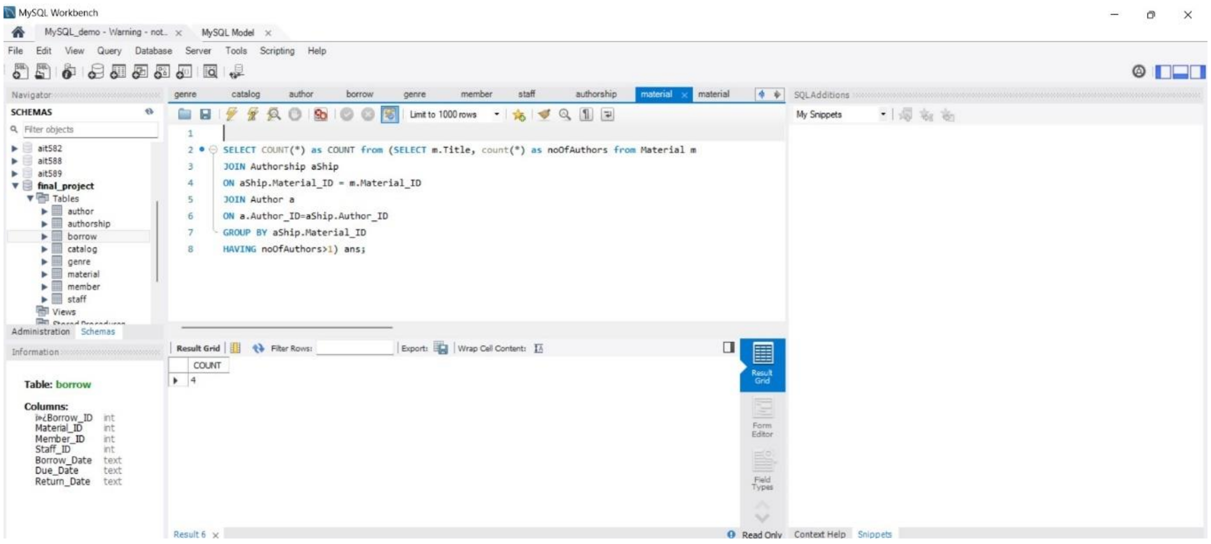
```
select * from author where Author_Id='20';

select * from author where Name ='Lucas Piki';
```

Author_ID	Name	Birth_Date	Nationality
20	Lucas Piki	1847-10-16	British
NULL	NULL	NULL	NULL

Query 5:

How many books were written by two or more authors?

SELECT COUNT(*) as COUNT from (SELECT m.Title, count(*) as noOfAuthors from Material m JOIN Authorship aShip ON aShip.Material_ID = m.Material_ID JOIN Author a ON a.Author_ID=aShip.Author_ID GROUP BY aShip.Material_ID HAVING noOfAuthors>1) ans;



```
SELECT COUNT(*) as COUNT from (SELECT m.Title, count(*) as noOfAuthors from Material m
JOIN Authorship aShip
ON aShip.Material_ID = m.Material_ID
JOIN Author a
ON a.Author_ID=aShip.Author_ID
GROUP BY aShip.Material_ID
HAVING noOfAuthors>1) ans;
```

COUNT
4

Table: borrow

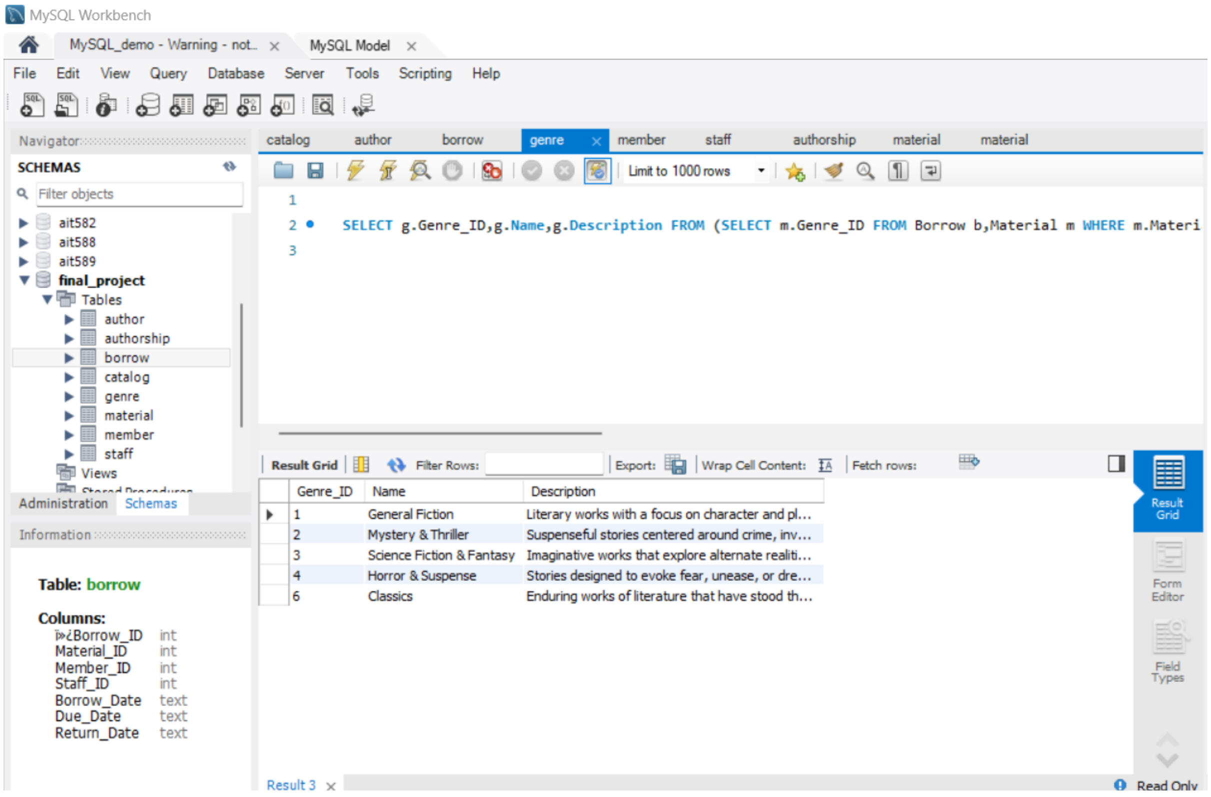
Columns:

ï»¿Borrow_ID	int
Material_ID	int
Member_ID	int
Staff_ID	int
Borrow_Date	text
Due_Date	text
Return_Date	text

Query 6:

What are the most popular genres in the library?

SELECT g.Genre_ID,g.Name,g.Description FROM (SELECT m.Genre_ID FROM Borrow b,Material m WHERE m.Material_ID=b.Material_ID) t,Genre g WHERE t.Genre_ID=g.Genre_ID GROUP BY g.Genre_ID ORDER BY g.Genre_ID ASC LIMIT 5;



```
SELECT g.Genre_ID,g.Name,g.Description FROM (SELECT m.Genre_ID FROM Borrow b,Material m WHERE m.Materi
```

Genre_ID	Name	Description
1	General Fiction	Literary works with a focus on character and pl...
2	Mystery & Thriller	Suspenseful stories centered around crime, inv...
3	Science Fiction & Fantasy	Imaginative works that explore alternate realiti...
4	Horror & Suspense	Stories designed to evoke fear, unease, or dre...
6	Classics	Enduring works of literature that have stood th...

Table: borrow

Columns:

ï»¿Borrow_ID	int
Material_ID	int
Member_ID	int
Staff_ID	int
Borrow_Date	text
Due_Date	text
Return_Date	text

Query 7:

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

SELECT COUNT(*) AS Count FROM

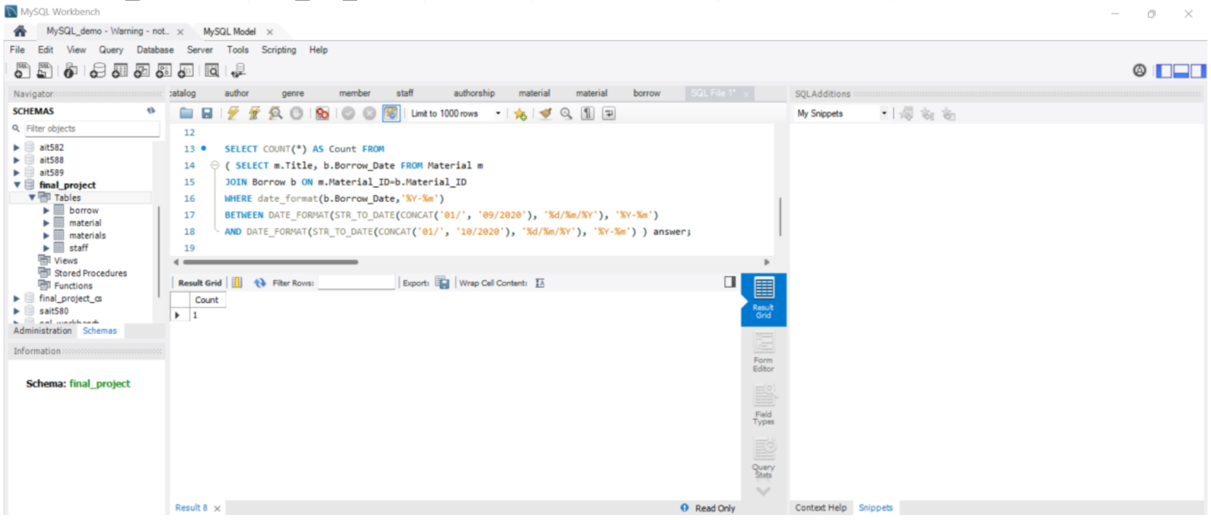
(SELECT m.Title, b.Borrow_Date FROM Material m

JOIN Borrow b ON m.Material_ID=b.Material_ID

WHERE date_format(b.Borrow_Date,'%Y-%m')

BETWEEN DATE_FORMAT(STR_TO_DATE(CONCAT('01/', '09/2020'), '%d/%m/%Y'), '%Y-%m')

AND DATE_FORMAT(STR_TO_DATE(CONCAT('01/', '10/2020'), '%d/%m/%Y'), '%Y-%m')) answer;



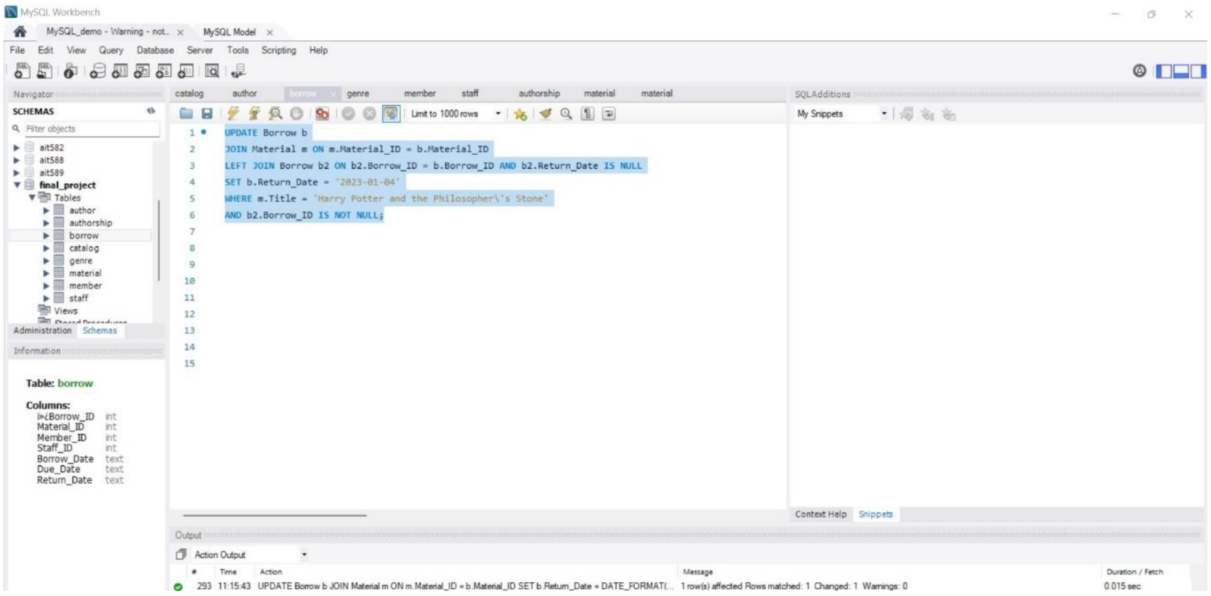
```
SELECT COUNT(*) AS Count FROM
( SELECT m.Title, b.Borrow_Date FROM Material m
JOIN Borrow b ON m.Material_ID=b.Material_ID
WHERE date_format(b.Borrow_Date,'%Y-%m')
BETWEEN DATE_FORMAT(STR_TO_DATE(CONCAT('01/', '09/2020'), '%d/%m/%Y'), '%Y-%m')
AND DATE_FORMAT(STR_TO_DATE(CONCAT('01/', '10/2020'), '%d/%m/%Y'), '%Y-%m') ) answer;
```

```
Count
1
```

Query 8:

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

UPDATE Borrow b JOIN Material m ON m.Material_ID = b.Material_ID SET b.Return_Date=DATE_FORMAT(STR_TO_DATE('04/01/2023', '%d/%m/%Y'), '%Y-%m-%d') WHERE m.Title='Harry Potter and the Philosopher\'s Stone' AND b2.Borrow_ID IS NOT NULL;



```sql
UPDATE Borrow b
JOIN Material m ON m.Material_ID = b.Material_ID
LEFT JOIN Borrow b2 ON b2.Borrow_ID = b.Borrow_ID AND b2.Return_Date IS NULL
SET b.Return_Date = '2023-01-04'
WHERE m.Title = 'Harry Potter and the Philosopher\'s Stone'
AND b2.Borrow_ID IS NOT NULL;
```

```
#    Time      Action                                                                                   Message                                                     Duration / Fetch
293  11:15:43  UPDATE Borrow b JOIN Material m ON m.Material_ID = b.Material_ID SET b.Return_Date = DATE_FORMAT(...  1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0  0.015 sec
```

Table: borrow

Columns:

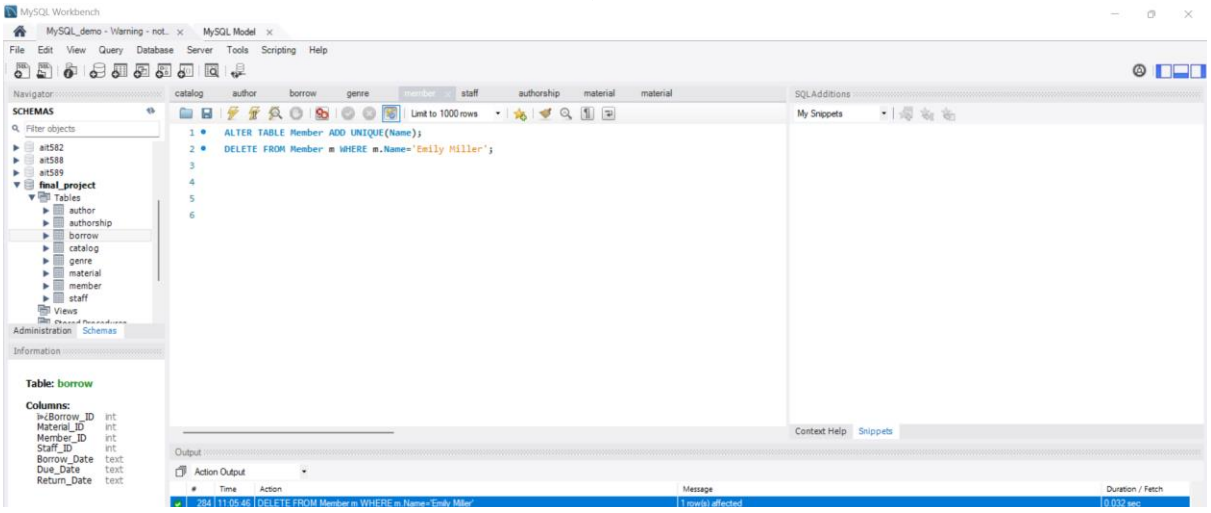
Borrow_ID	int
Material_ID	int
Member_ID	int
Staff_ID	int
Borrow_Date	text
Due_Date	text
Return_Date	text

Query 9:

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

ALTER TABLE Member ADD UNIQUE(Name);

DELETE FROM Member m WHERE m.Name='Emily Miller';



```sql
ALTER TABLE Member ADD UNIQUE(Name);
DELETE FROM Member m WHERE m.Name='Emily Miller';
```

```
#    Time      Action                                            Message            Duration / Fetch
284  11:05:46  DELETE FROM Member m WHERE m.Name='Emily Miller'  1 row(s) affected  0.032 sec
```

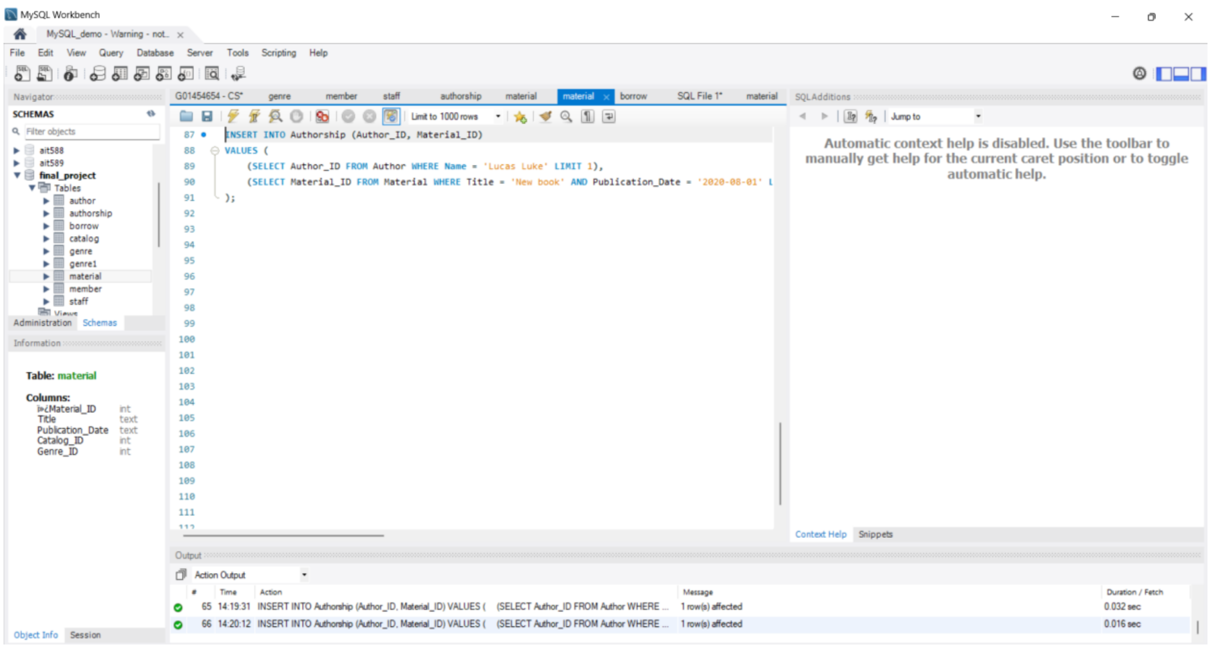
Table: borrow

Columns:

Borrow_ID	int
Material_ID	int
Member_ID	int
Staff_ID	int
Borrow_Date	text
Due_Date	text
Return_Date	text

Query 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 Authorship (Author_ID, Material_ID)
VALUES (
   (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' LIMIT 1)
);
```



```
INSERT INTO Authorship (Author_ID, Material_ID)
VALUES (
    (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' L
);
```

```
#   Time      Action                                                                                    Message            Duration / Fetch
65  14:19:31  INSERT INTO Authorship (Author_ID, Material_ID) VALUES (  (SELECT Author_ID FROM Author WHERE ...  1 row(s) affected  0.032 sec
66  14:20:12  INSERT INTO Authorship (Author_ID, Material_ID) VALUES (  (SELECT Author_ID FROM Author WHERE ...  1 row(s) affected  0.016 sec
```

FUTURE WORK:

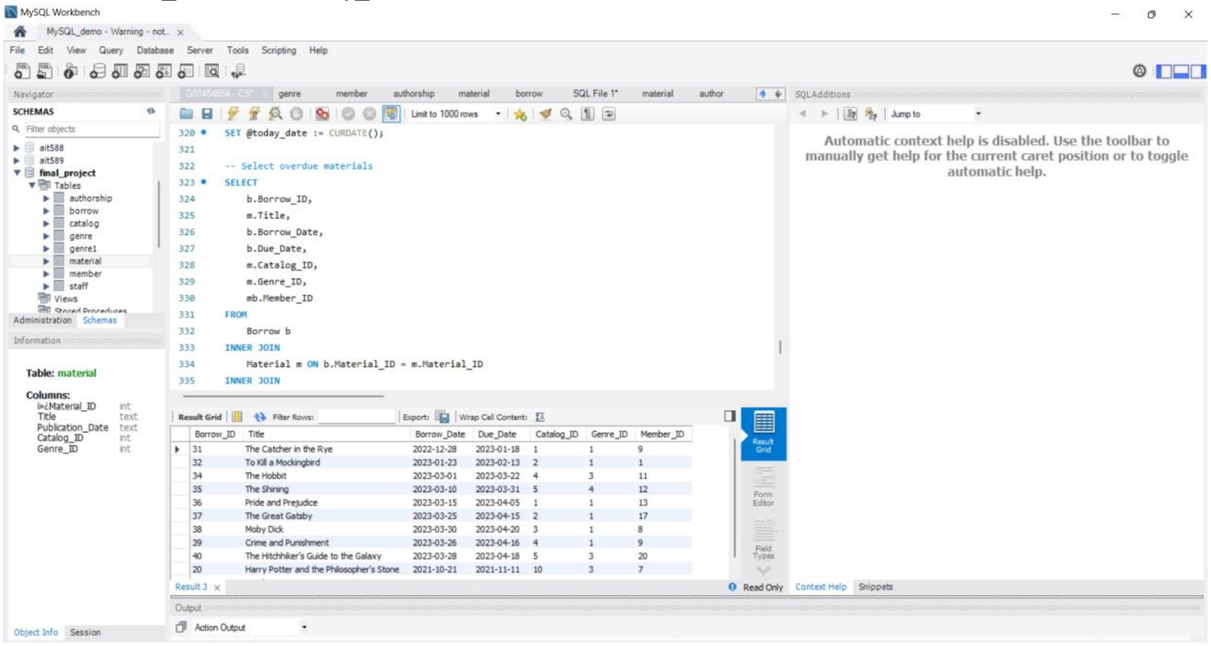
Alert staff about overdue materials on a daily basis?

When a new borrow deadline has passed, staff members can receive a message. Additionally, past due books can be checked daily to update their status as due books. The staff may see which books are due on a specific date and which materials are past due once they register into their system.

Query that should be run for every 24hrs:

```
SET @today_date := CURDATE();
SELECT
   b.Borrow_ID,
   m.Title,
   b.Borrow_Date,
   b.Due_Date,
   m.Catalog_ID,
   m.Genre_ID,
   mb.Member_ID
FROM
   Borrow b
INNER JOIN
```

```
    Material m ON b.Material_ID = m.Material_ID
INNER JOIN
    Member mb ON b.Member_ID = mb.Member_ID
WHERE
    b.Return_Date IS NULL
    AND b.Due_Date < @today_date;
```



```
SET @today_date := CURDATE();

-- Select overdue materials
SELECT
    b.Borrow_ID,
    m.Title,
    b.Borrow_Date,
    b.Due_Date,
    m.Catalog_ID,
    m.Genre_ID,
    mb.Member_ID
FROM
    Borrow b
INNER JOIN
    Material m ON b.Material_ID = m.Material_ID
INNER JOIN
```

Borrow_ID	Title	Borrow_Date	Due_Date	Catalog_ID	Genre_ID	Member_ID
31	The Catcher in the Rye	2022-12-28	2023-01-18	1	1	9
32	To Kill a Mockingbird	2023-01-23	2023-02-13	2	1	1
34	The Hobbit	2023-03-01	2023-03-22	4	3	11
35	The Shining	2023-03-10	2023-03-31	5	4	12
36	Pride and Prejudice	2023-03-15	2023-04-05	1	1	13
37	The Great Gatsby	2023-03-25	2023-04-15	2	1	17
38	Moby Dick	2023-03-30	2023-04-20	3	1	8
39	Crime and Punishment	2023-03-26	2023-04-16	4	1	9
40	The Hitchhiker's Guide to the Galaxy	2023-03-28	2023-04-18	5	3	20
20	Harry Potter and the Philosopher's Stone	2021-10-21	2021-11-11	10	3	7

Table: material

Columns:

Material_ID	int
Title	text
Publication_Date	text
Catalog_ID	int
Genre_ID	int

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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 add a new column to the member table for deactivation. Active or Inactive would be the values for the Status column. Only when the status is active the books can be borrowed. When the status is inactive, borrowing can be halted. We must tally the quantity of past-due books in order to switch between the active and inactive status. We will make a new table called Overdue table with the columns Material_ID, Member_ID, and Due_Date in order to handle this. The material ID, member ID, and due date of a book will be added to the Overdue database once it has passed the Due_Date. We will add a trigger to the Overdue table for insertion and deletion so that the status in the Member table can be toggled.

Every time we add a new record, we count the number of past-due items for that member. If the count is three or more, the member's status will be changed to inactive. Similarly, when we remove a record, we count the past-due items once more. If the count is three or less, the member's status will be changed to active.

GUIDELINES TO RUN THE CODE:

1. First create a database and use the database by performing sql queries.
2. Then create all the tables using the SQL queries.
3. Insert all the values into the tables.
4. After insertion execute the queries for the given questions.

CONCLUSION:

In conclusion, in order to facilitate the management of the library resources that members borrow and store, I have created a Library Management System. The primary keys and foreign keys that I have used will help to keep the database consistent. I have integrated DDL (Data Definition Language) and DML (Data Manipulation Language) in this by using MySQL for the creation, insertion, and updating of values. Hence, the library management system is created using SQL.