# LIBRARY MANAGEMENT

CS-504-003 PROJECT

PROF. YIN BINQIAN
GEORGE MASON UNIVERSITY

**Atharva Abhay Shrikhande**
**G01462301**

## Introduction –

In the fast-paced world of retail book sales, understanding the complexities of customer behavior and sales profitability is critical for maintaining business growth and optimizing operational strategies. This project examines the detailed financial landscape of a bookstore's transactions, with a focus on determining the total profit generated by each customer. By combining data from multiple touchpoints—customer profiles, order details, and book inventory—our analysis aims to reveal critical insights that will improve decision-making processes and foster customer relationships.

The primary goal of this analytical endeavor is to determine the total profit derived from each customer, considering the number of books purchased and the cost dynamics associated with each sale. This entails a thorough examination of sales data linked to several relational database tables, such as Customers, Orders, OrderItems, and Books. Each table contributes significantly to the overall picture of the bookstore's operational efficiency and customer purchasing patterns.

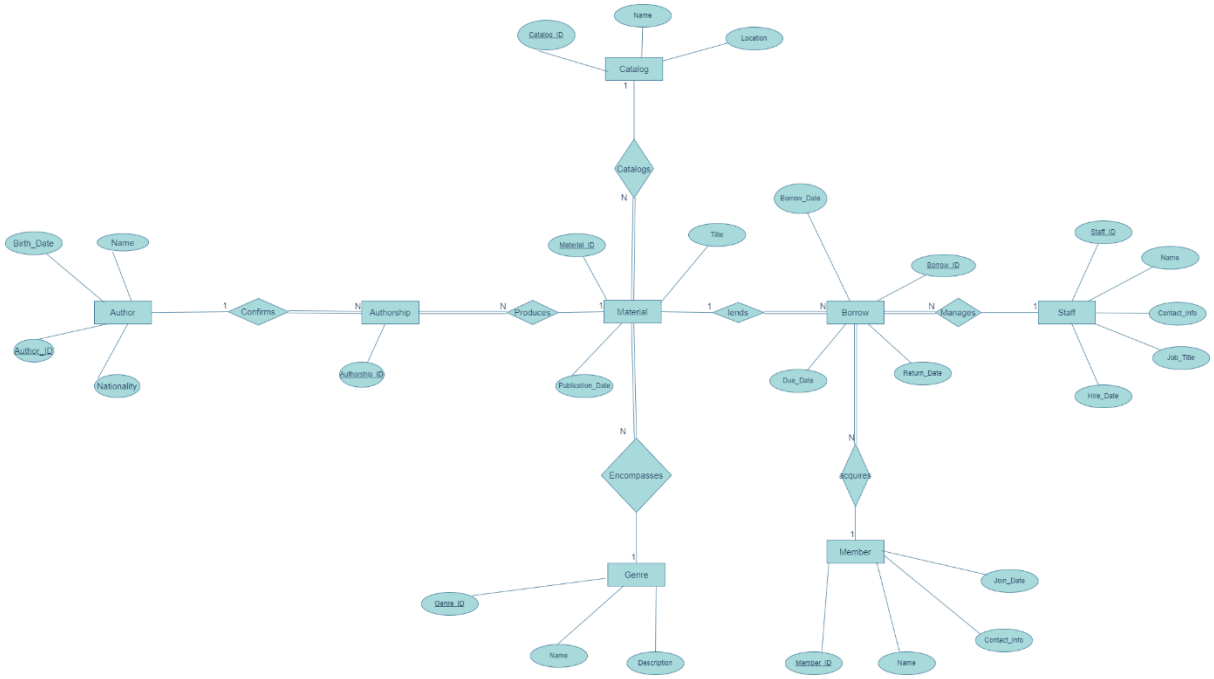
## Scope of the Project -

The aim of this project is to create a cutting-edge library management system for a public library, designed to replace traditional management methods with a highly efficient and error-free relational database. This system will allow for seamless tracking and management of library resources, improving user experience and operational efficiency. The system will be organized around eight main entities: Material, Borrow, Author, Authorship, Catalog, Genre, Member, and Staff, each with its own set of attributes that capture the critical information required for comprehensive resource management.

An Entity-Relationship (ER) Diagram will be used to visually map the relationships between these entities, ensuring consistency in database structure and interaction. This approach not only allows for a more intuitive understanding of the database's architecture, but it also helps to maintain data integrity and consistency across multiple library functions such as material lending, returns, cataloging, and member management.

By implementing this system, the library hopes to significantly reduce administrative burdens, improve data accessibility and reliability, and provide a scalable solution that can adapt to future needs and technologies. The scope of this project includes designing, developing, and deploying the database system to ensure a smooth transition and effective use of the new system.

## ERD structure is as follows –



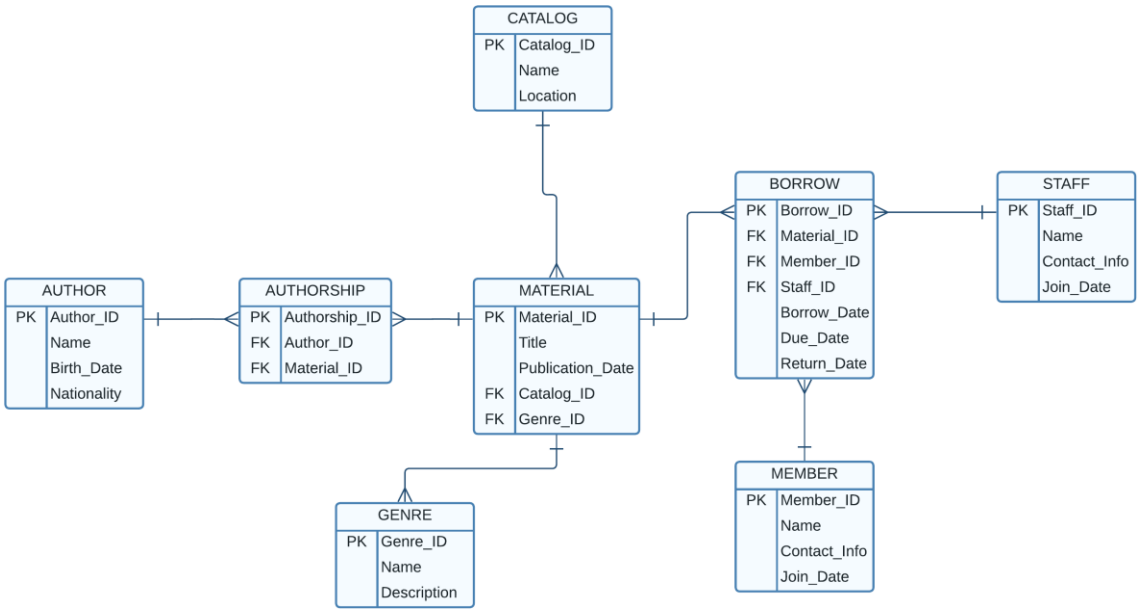

## Relationships :-

In Entity-Relationship Diagrams (ERDs), relationships are used to depict how various data points are related. One-to-many (1:M) relationships mean that one record in one entity can be linked to multiple records in another, indicating a single source that branches out. Many-to-many (M:M) relationships indicate that records in one entity can relate to many in another, and vice versa, which frequently necessitates the use of an additional table to manage these links effectively.

In this ERD, the author-to-author relationship is 1:M, and the author-to-material relationship is M:1. The author-to-material relationship is only M:N. Material to borrow is 1:M, Member to borrow is 1:M, Staff to borrow is 1:M, Material to Catalog is 1:M, and Material to Genre is M:1.

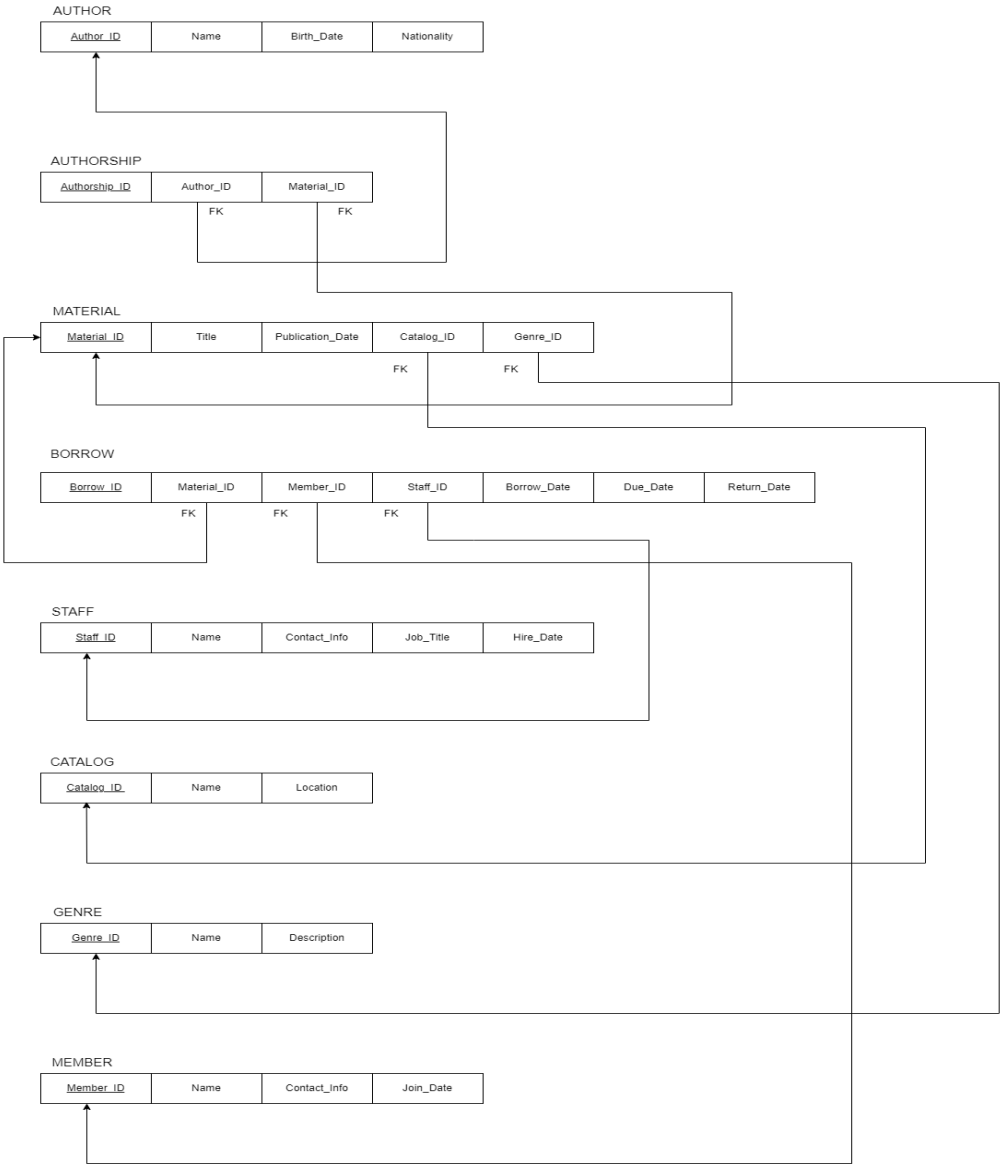
## Cardinality Ratios:-

Cardinality ratios in Entity-Relationship Diagrams (ERDs) help depict the kind of relationship between entities by specifying the number of instances in one entity that can or must be associated with the number of instances in another. The 1:M cardinality indicates that a single instance of one entity can be linked to multiple instances of another entity. The M:M cardinality indicates that multiple instances of one entity can interact with multiple instances of another, which is typically managed by an intermediary table. Each ratio contributes to defining the rules governing how entities interact in a database.




For this project, I am going to utilize PostgreSQL and Valentina database and for ERD and schema I have used draw.io and Lucid Chart.

# The relational schema is as follows:-



## DDL Commands :-

Data Definition Language (DDL) is a set of SQL commands used to define and modify the structure of database objects such as tables, schemas, indexes, and database links. CREATE, ALTER, and DROP are common DDL commands for creating, modifying, and deleting database structures, respectively. DDL allows the specification of data types, structures, and constraints, ensuring that the data follows the rules defined by the database administrator.

```
--TABLE CREATION--

CREATE TABLE Author (
Author_ID NUMERIC PRIMARY KEY,
NAME VARCHAR(60) NOT NULL,
Birth_Date DATE,
Nationality VARCHAR(20)
);

CREATE TABLE "Catalog"(
Catalog_ID NUMERIC PRIMARY KEY,
NAME VARCHAR(255) NOT NULL,
LOCATION VARCHAR (50));

CREATE TABLE Genre(
Genre_ID NUMERIC PRIMARY KEY,
NAME VARCHAR(255) NOT NULL,
Description VARCHAR(500));

CREATE TABLE Material (
Material_ID NUMERIC PRIMARY KEY,
Title VARCHAR(255) NOT NULL,
Publication_date DATE,
Catalog_ID NUMERIC REFERENCES "Catalog"(Catalog_ID),
Genre_ID NUMERIC REFERENCES Genre(Genre_ID)
);

CREATE TABLE Authorship(
Authorship_ID NUMERIC PRIMARY KEY,
Author_ID NUMERIC REFERENCES Author(Author_ID),
Material_id NUMERIC REFERENCES Material(Material_ID));

CREATE TABLE Member(
Member_ID NUMERIC PRIMARY KEY,
NAME VARCHAR(255) NOT NULL,
Contact VARCHAR(255) NOT NULL,
Join_Date DATE);
```

```

CREATE TABLE Staff(
Staff_ID NUMERIC PRIMARY KEY,
NAME VARCHAR(255) NOT NULL,
Contact_Info VARCHAR(255) NOT NULL,
Job_Title VARCHAR(255) NOT NULL,
Hire_Date DATE);

CREATE TABLE Borrow(
Borrow_ID NUMERIC PRIMARY KEY,
Borrow_Date Date,
Due_Date DATE,
Return_Date DATE,
Material_ID NUMERIC REFERENCES Material(Material_ID),
Member_ID NUMERIC REFERENCES Member(Member_ID),
Staff_ID NUMERIC REFERENCES Staff(Staff_ID));
```

## DML Commands:-

Data Manipulation Language (DML) is a subset of SQL that manages data within database objects like tables. Common DML commands are INSERT, UPDATE, DELETE, and SELECT. These commands allow you to add new rows, modify existing data, remove rows, and query database tables. DML is essential for day-to-day data manipulation, as it allows you to handle data in a way that supports applications and user interactions.

```sql
--INSERTING VALUES--

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');


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');
```

```sql
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.');


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');


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
INSERT INTO Material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID)
VALUES
(1, 'The Catcher in the Rye', '1951-07-16', 1, 1),
(2, 'To Kill a Mockingbird', '1960-07-11', 2, 1),
(3, 'The Da Vinci Code', '2003-04-01', 3, 2),
(4, 'The Hobbit', '1937-09-21', 4, 3),
(5, 'The Shining', '1977-01-28', 5, 4),
(6, 'Pride and Prejudice', '1813-01-28', 1, 1),
(7, 'The Great Gatsby', '1925-04-10', 2, 1),
(8, 'Moby Dick', '1851-10-18', 3, 1),
(9, 'Crime and Punishment', '1866-01-01', 4, 1),
(10, 'The Hitchhiker''s Guide to the Galaxy', '1979-10-12', 5, 3),
(11, '1984', '1949-06-08', 1, 5),
(12, 'Animal Farm', '1945-08-17', 2, 5),
(13, 'The Haunting of Hill House', '1959-10-17', 3, 4),
(14, 'Brave New World', '1932-08-01', 4, 5),
(15, 'The Chronicles of Narnia: The Lion the Witch and the Wardrobe', '1950-10-16', 5, 3),
(16, 'The Adventures of Huckleberry Finn', '1884-12-10', 6, 1),
(17, '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);


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', '2019-09-26'),
(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'),
```

```
(21, 21, 1, 3, '2021-11-29', '2021-12-20', NULL),
(22, 22, 2, 3, '2022-01-10', '2022-01-21', '2022-01-25'),
(23, 23, 3, 3, '2022-02-07', '2022-02-25', '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', '2022-05-15'),
(26, 26, 6, 3, '2022-06-22', '2022-07-13', '2022-07-05'),
(27, 27, 7, 3, '2022-08-04', '2022-08-25', '2022-08-23'),
(28, 28, 8, 3, '2022-09-13', '2022-10-04', '2022-09-26'),
(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, 12, 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);


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);
```

# Tables:-

## (1) Author Table:-

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

## (2) Authorship Table:-

| | authorship_id | author_id | material_id |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 |
| 3 | 3 | 3 | 3 |
| 4 | 4 | 4 | 4 |
| 5 | 5 | 5 | 5 |
| 6 | 6 | 6 | 6 |
| 7 | 7 | 7 | 7 |
| 8 | 8 | 8 | 8 |
| 9 | 9 | 9 | 9 |

Number of records: 35 Number of fields: 3 Query time: 0 millisecond(s) Read-Only

## (3) Borrow Table:-

| | borrow_id | borrow_date | due_date | return_date | material_id | member_id | staff_id |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 2018-09-12 | 2018-10-03 | 2018-09-30 | 1 | 1 | 1 |
| 2 | 2 | 2018-10-15 | 2018-11-05 | 2018-10-29 | 2 | 2 | 1 |
| 3 | 3 | 2018-12-20 | 2019-01-10 | 2019-01-08 | 3 | 3 | 1 |
| 4 | 4 | 2019-03-11 | 2019-04-01 | 2019-03-27 | 4 | 4 | 1 |
| 5 | 6 | 2019-07-05 | 2019-07-26 | 2019-07-21 | 6 | 6 | 1 |
| 6 | 7 | 2019-09-10 | 2019-10-01 | 2019-09-25 | 7 | 7 | 1 |
| 7 | 8 | 2019-11-08 | 2019-11-29 | 2019-11-20 | 8 | 8 | 1 |
| 8 | 9 | 2020-01-15 | 2020-02-05 | 2020-02-03 | 9 | 9 | 1 |
| 9 | 10 | 2020-03-12 | 2020-04-02 | 2020-03-28 | 10 | 10 | 1 |
| 10 | 11 | 2020-05-14 | 2020-06-04 | 2020-05-28 | 1 | 11 | 2 |
| 11 | 12 | 2020-07-21 | 2020-08-11 | 2020-08-02 | 2 | 12 | 2 |
| 12 | 13 | 2020-09-25 | 2020-10-16 | 2020-10-15 | 3 | 13 | 2 |
| 13 | 14 | 2020-11-08 | 2020-11-29 | 2020-11-24 | 4 | 1 | 2 |
| 14 | 15 | 2021-01-03 | 2021-01-24 | 2021-01-19 | 5 | 2 | 2 |
| 15 | 16 | 2021-02-18 | 2021-03-11 | 2021-03-12 | 6 | 3 | 2 |
| 16 | 17 | 2021-04-27 | 2021-05-18 | 2021-05-20 | 17 | 4 | 2 |
| 17 | 19 | 2021-08-15 | 2021-09-05 | 2021-09-03 | 19 | 6 | 2 |
| 18 | 21 | 2021-11-29 | 2021-12-20 | *<NULL>* | 21 | 1 | 3 |
| 19 | 22 | 2022-01-10 | 2022-01-31 | 2022-01-25 | 22 | 2 | 3 |

Number of records: 37 Number of fields: 7 Query time: 8 millisecond(s) Read-Only

## (4) Catalog Table:-

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

## (5) Genre Table:-

| | genre_id | name | description |
|---|---|---|---|
| 1 | 1 | General Fiction | Literary works with a focus on character and plot development, exp... |
| 2 | 2 | Mystery & Thriller | Suspenseful stories centered around crime, investigation, or espion... |
| 3 | 3 | Science Fiction & Fantasy | Imaginative works that explore alternate realities, futuristic concept... |
| 4 | 4 | Horror & Suspense | Stories designed to evoke fear, unease, or dread, often featuring su... |
| 5 | 6 | Classics | Enduring works of literature that have stood the test of time, often ... |
| 6 | 5 | Dystopian & Apocalyptic | Depictions of societies in decline or collapse, often exploring them... |
| 7 | 7 | Historical Fiction | Fictional stories set in the past, often based on real historical event... |
| 8 | 8 | Epic Poetry & Mythology | Ancient or traditional stories and poems, often featuring heroes, go... |

## (6) Material Table:-

| | material_id | title | publication_date | catalog_id | genre_id |
|---|---|---|---|---|---|
| 1 | 1 | The Catcher in the Rye | 1951-07-16 | 1 | 1 |
| 2 | 2 | To Kill a Mockingbird | 1960-07-11 | 2 | 1 |
| 3 | 3 | The Da Vinci Code | 2003-04-01 | 3 | 2 |
| 4 | 4 | The Hobbit | 1937-09-21 | 4 | 3 |
| 5 | 5 | The Shining | 1977-01-28 | 5 | 4 |
| 6 | 6 | Pride and Prejudice | 1813-01-28 | 1 | 1 |
| 7 | 7 | The Great Gatsby | 1925-04-10 | 2 | 1 |
| 8 | 8 | Moby Dick | 1851-10-18 | 3 | 1 |
| 9 | 9 | Crime and Punishment | 1866-01-01 | 4 | 1 |
| 10 | 10 | The Hitchhiker's Guide to the Galaxy | 1979-10-12 | 5 | 3 |
| 11 | 11 | 1984 | 1949-06-08 | 1 | 5 |
| 12 | 12 | Animal Farm | 1945-08-17 | 2 | 5 |
| 13 | 13 | The Haunting of Hill House | 1959-10-17 | 3 | 4 |
| 14 | 14 | Brave New World | 1932-08-01 | 4 | 5 |
| 15 | 15 | The Chronicles of Narnia: The Lion the Witch and the Wardrobe | 1950-10-16 | 5 | 3 |
| 16 | 16 | The Adventures of Huckleberry Finn | 1884-12-10 | 6 | 1 |
| 17 | 17 | The Catch-22 | 1961-10-11 | 7 | 1 |
| 18 | 18 | The Picture of Dorian Gray | 1890-07-01 | 8 | 1 |
| 19 | 19 | The Call of Cthulhu | 1928-02-01 | 9 | 4 |

Number of records: 32 Number of fields: 5 Query time: 2 millisecond(s) Read-Only

## (7) Member Table:-

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

## (8) Staff Table:-

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

# Queries:-

## (1) Which materials are currently available in the library? If a material is borrowed and not

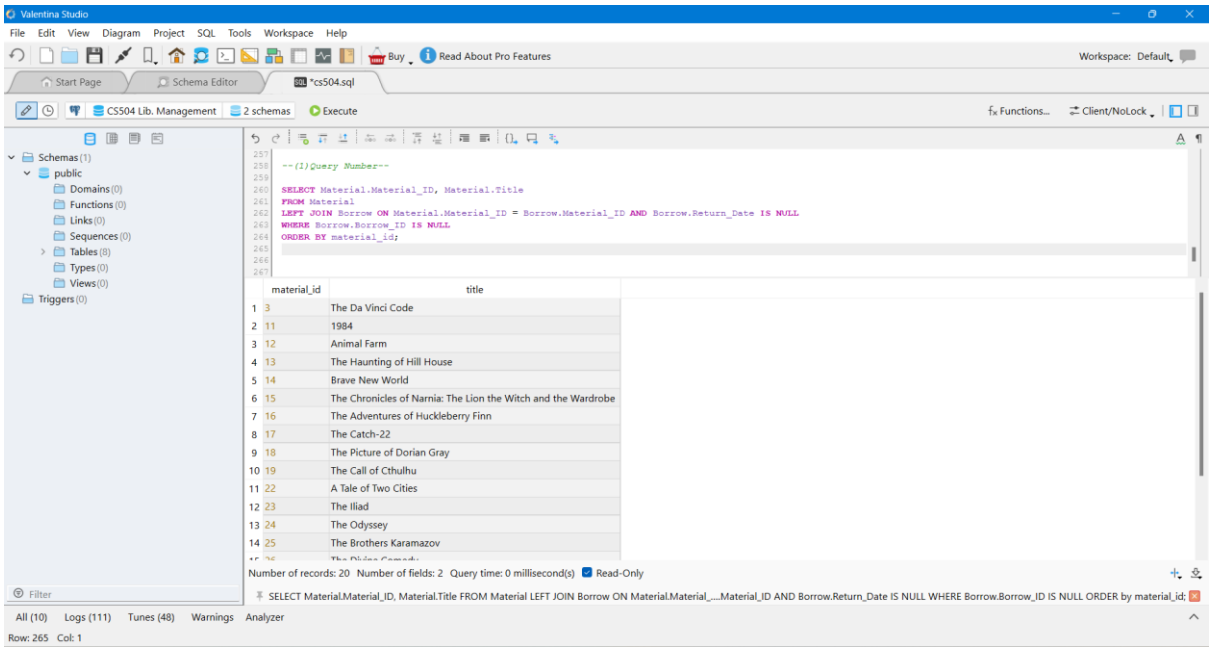
## returned, it's not considered as available.

**Code-**

```
SELECT Material.Material_ID, Material.Title
FROM Material
LEFT JOIN Borrow ON Material.Material_ID = Borrow.Material_ID AND Borrow.Return_Date IS NULL
WHERE Borrow.Borrow_ID IS NULL
ORDER by material_id;
```

**Explanation-**

The SQL query uses a LEFT JOIN to combine the Material and Borrow tables, and this will pull records that contain materials that are still borrowed (Return_Date is NULL). Then, it will filter for materials that don't have active borrow records, whereby they list all materials that are available for borrowing since there are either returns or they've never been checked out.



**(2) Which materials are currently overdue? Suppose today is 04/01/2023, and show the**

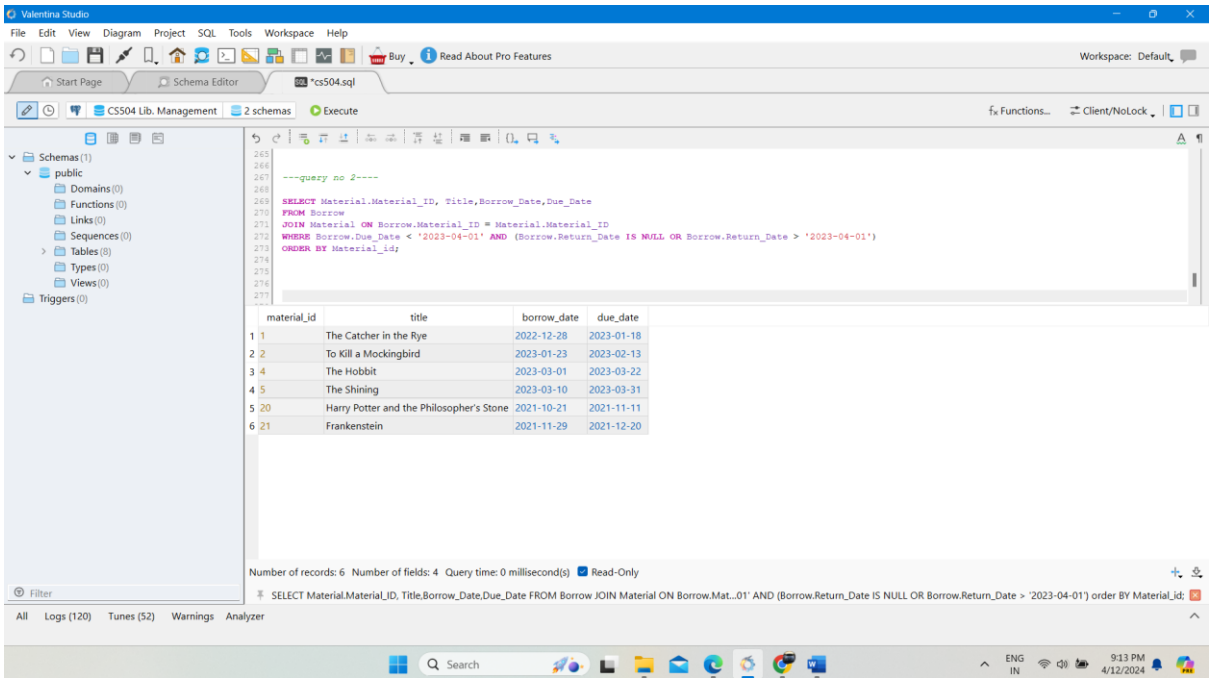
**borrow date and due date of each material.**

**Code –**

```
SELECT Material.Material_ID, Title,Borrow_Date,Due_Date
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
WHERE Borrow.Due_Date < '2023-04-01' AND (Borrow.Return_Date IS NULL OR Borrow.Return_Date > '2023-04-01')
order BY Material_id;
```

**Explanation –**

In this query, I am joining both tables, Material and Borrow, based on Material_ID with an implication to connect every borrowed item to its descriptive title in the Material table. What I need to do is get the borrow and due date from the Borrow table, along with the title of each material from the Material table, so as to get a full view of every overdue item. This query is looking for materials whose due date was prior to April 1, 2023, either not yet returned or returned after the due date, indicating all overdue materials.



**(3) What are the top 10 most borrowed materials in the library? Show the title of each.**

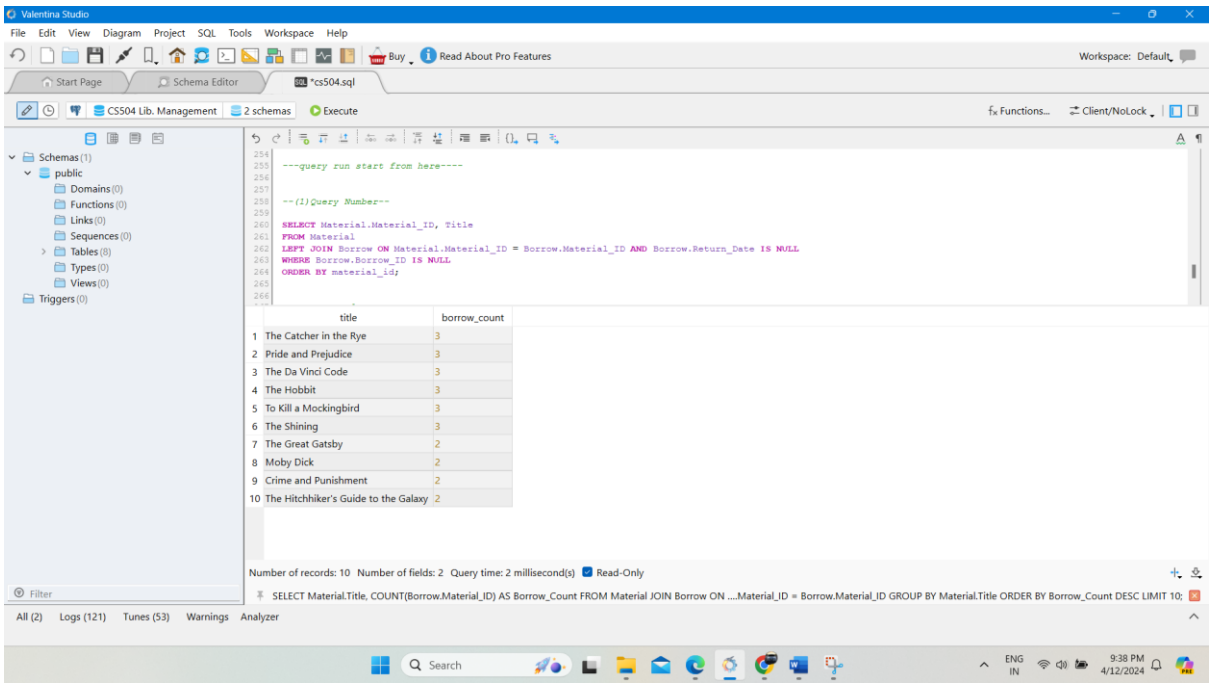
**material and order them based on their available counts.**

**Code-**

```
SELECT Material.Title, COUNT(Borrow.Material_ID) AS Borrow_Count
FROM Material
JOIN Borrow ON Material.Material_ID = Borrow.Material_ID
GROUP BY Material.Title
ORDER BY Borrow_Count DESC
LIMIT 10;
```

**Explanation –**

This query links both the Material and Borrow tables, and then associates each borrow record with its material title. Once that is established, it uses the COUNT function to sum the total number of borrows for each material, with the grouping of results being made by material title to ensure that each title is counted once only. The results are then sorted in descending order by borrow count, and only the top ten materials are displayed, accentuating the library's most popular titles.



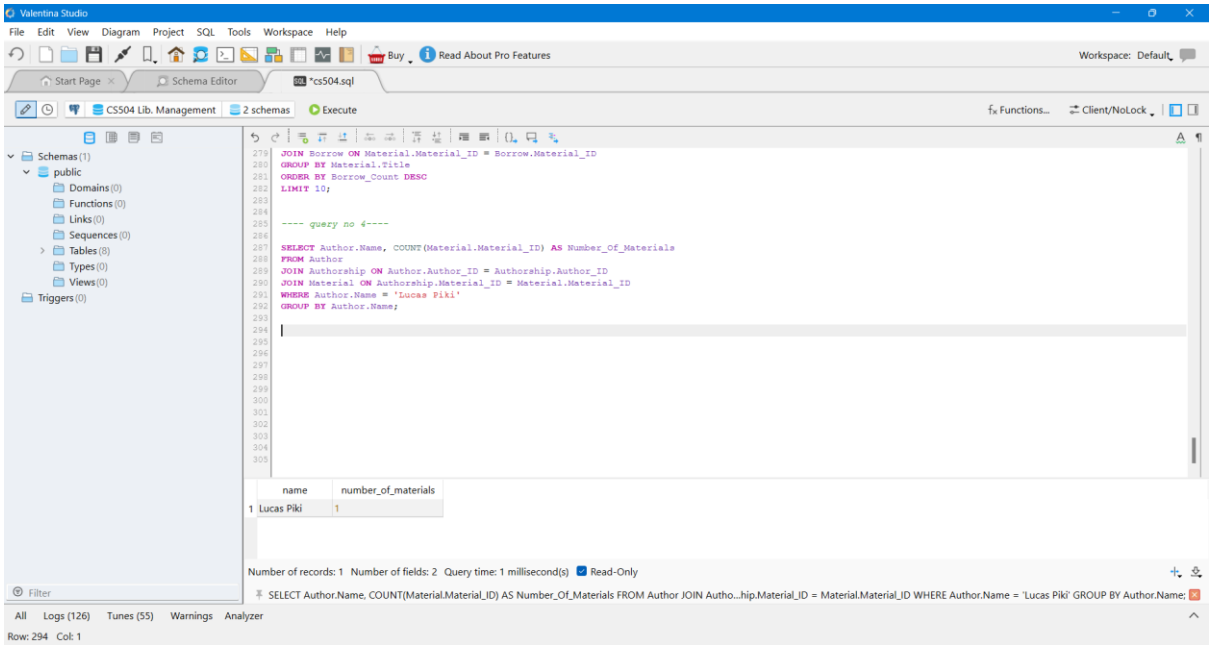
## (4) How many materials has the author Lucas Piki written?

**Code –**

```
SELECT Author.Name, COUNT(Material.Material_ID) AS Number_Of_Materials
FROM Author
JOIN Authorship ON Author.Author_ID = Authorship.Author_ID
JOIN Material ON Authorship.Material_ID = Material.Material_ID
WHERE Author.Name = 'Lucas Piki'
GROUP BY Author.Name;
```

**Explanation –**

The query starts by connecting the Author, Authorship, and Material tables to ensure that all relevant information about the author, authorship records, and materials are in sync. By including Lucas Piki in the WHERE clause, the query limits it to materials that are related to this author. GROUP BY AUTHOR.Name enables author-based aggregation.



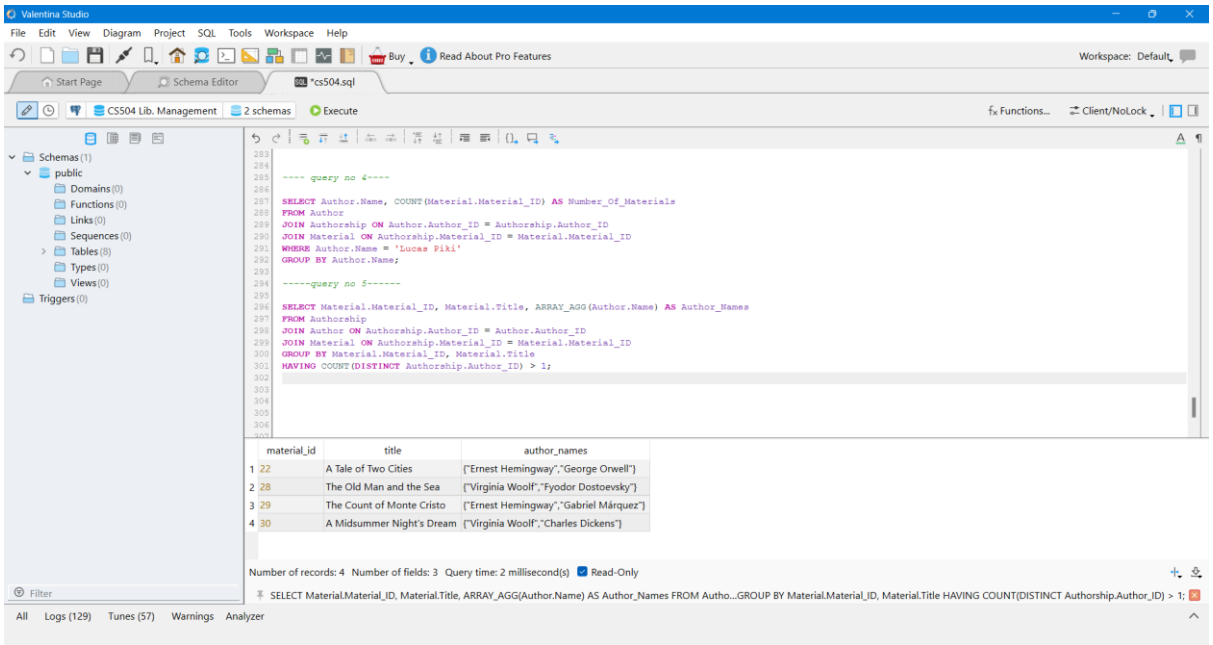
## (5) How many materials were written by two or more authors?

**Code-**

```
SELECT Material.Material_ID, Material.Title, ARRAY_AGG(Author.Name) AS Author_Names
FROM Authorship
JOIN Author ON Authorship.Author_ID = Author.Author_ID
JOIN Material ON Authorship.Material_ID = Material.Material_ID
GROUP BY Material.Material_ID, Material.Title
HAVING COUNT(DISTINCT Authorship.Author_ID) > 1;
```

**Explanation-**

The JOIN operation joins the Authorship table to the Author and Material tables. This ensures that the names of authors, found in the Author table (Author.Name), and material titles can be fetched. The GROUP BY Clause sorts the results by Material.Material_ID and material. The title, which shall be used to group the author names for each unique material. The ARRAY_AGG function collects all author names into an array for every grouped material, representing which authors collaborated with each work. This function is PostgreSQL-specific. The HAVING clause limits the groups to materials that have more than one distinct author, which is essential for the process of identifying collaborations.



## (6) What are the most popular genres in the library ranked by the total number of borrowed.

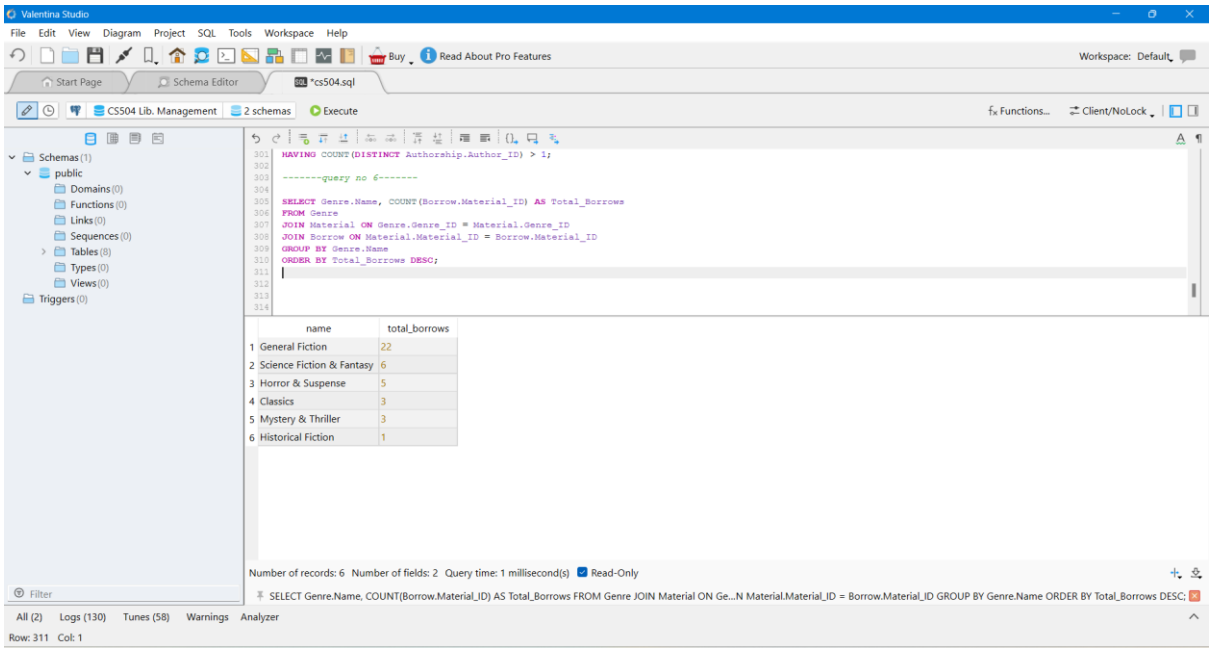
## times of each genre?

**Code –**

```
SELECT Genre.Name, COUNT(Borrow.Material_ID) AS Total_Borrows
FROM Genre
JOIN Material ON Genre.Genre_ID = Material.Genre_ID
JOIN Borrow ON Material.Material_ID = Borrow.Material_ID
GROUP BY Genre.Name
ORDER BY Total_Borrows DESC;
```

**Explanation –**

For this query, I'm using Join, which links the Material and Genre tables by linking materials to genres through Genre_ID. Join Borrow ON, which connects the Borrow and Material tables, tracks how many times each material has been borrowed. GROUP BY Genre.Name is used to group the results by genre name so the COUNT aggregation function can calculate the amount of borrowed material within a genre. Finally, ORDER BY Total_Borrows DESC orders the results in descending order by total borrows, simply sorting from most to least popular based on borrowing activity.



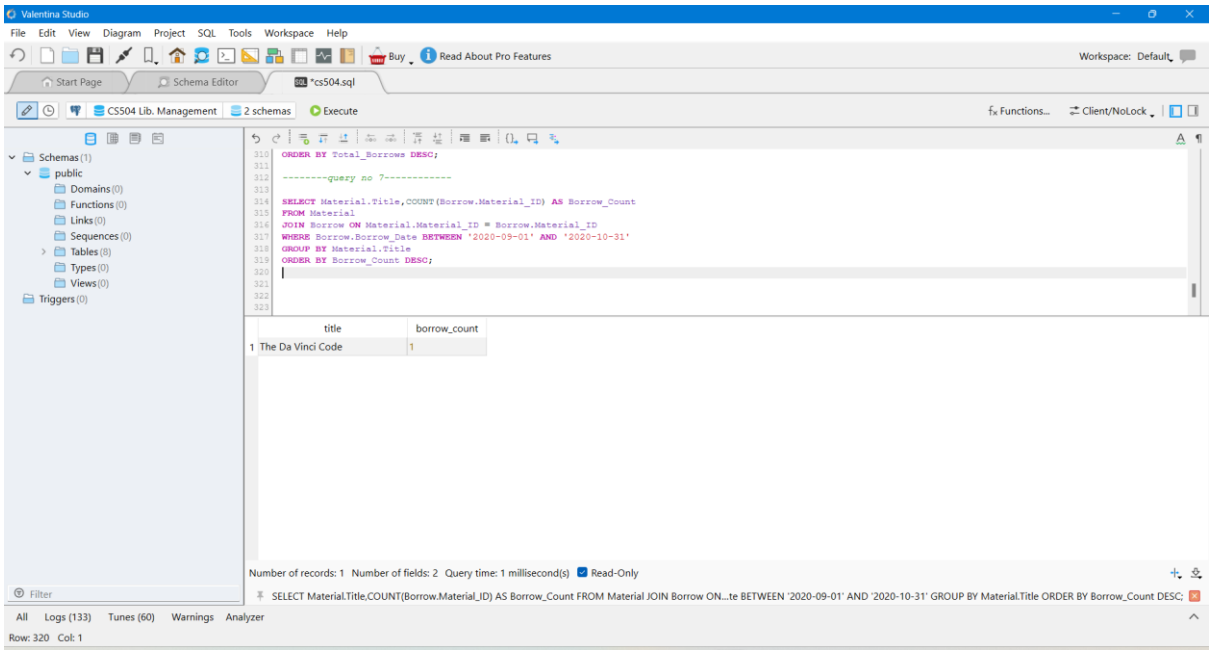
## (7) How many materials had been borrowed from 09/2020-10/2020?

**Code-**

```
SELECT Material.Title,COUNT(Borrow.Material_ID) AS Borrow_Count
FROM Material
JOIN Borrow ON Material.Material_ID = Borrow.Material_ID
WHERE Borrow.Borrow_Date BETWEEN '2020-09-01' AND '2020-10-31'
GROUP BY Material.Title
ORDER BY Borrow_Count DESC;
```

**Explanation –**

The query examines the borrowing records from this time period. It identifies each item by name, counts how many times it has been borrowed, and returns the total number of items available in the library. To ensure accurate aggregation, the results are grouped by material and then sorted by number of borrowings, with the most frequently borrowed materials highlighted first.



## (8) How do you update the “Harry Po er and the Philosopher's Stone” when it is returned on

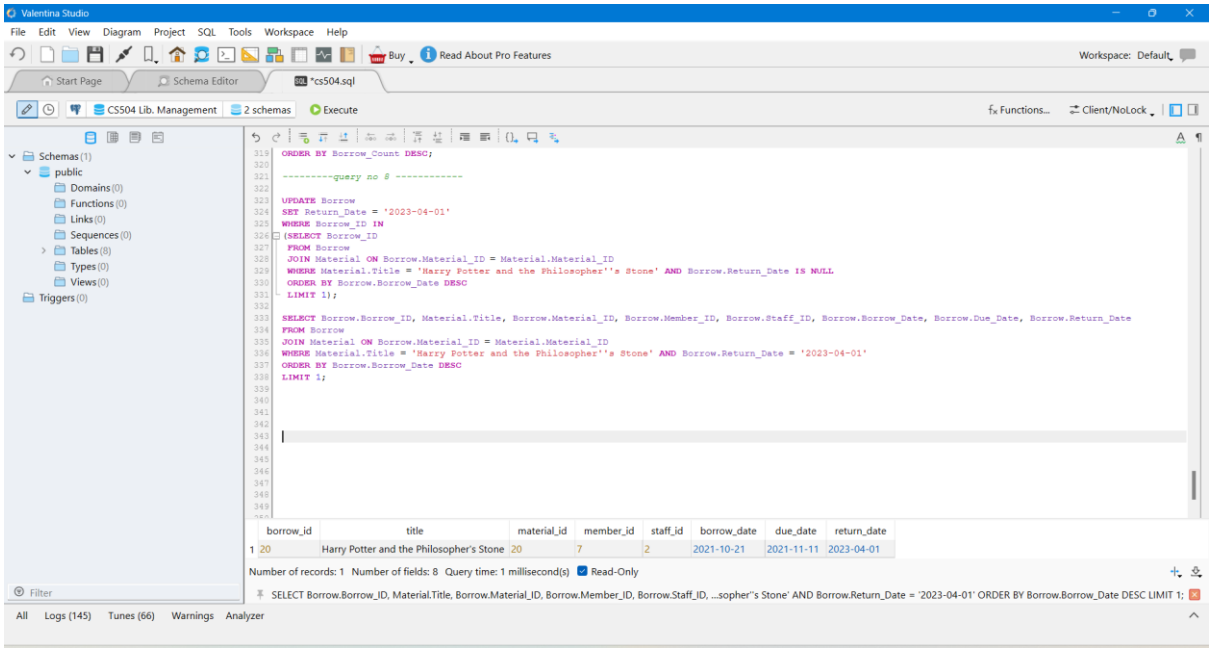
## 04/01/2023?

**Code-**

```
UPDATE Borrow

SET Return_Date = '2023-04-01'

WHERE Borrow_ID IN

(SELECT Borrow_ID

 FROM Borrow

 JOIN Material ON Borrow.Material_ID = Material.Material_ID

 WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return_Date IS NULL

 ORDER BY Borrow.Borrow_Date DESC

 LIMIT 1);


SELECT Borrow.Borrow_ID, Material.Title, Borrow.Material_ID, Borrow.Member_ID, Borrow.Staff_ID, Borrow.Borrow_Date, Borrow.Due_Date, Borrow.Return_Date

FROM Borrow

JOIN Material ON Borrow.Material_ID = Material.Material_ID

WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return_Date = '2023-04-01'

ORDER BY Borrow.Borrow_Date DESC

LIMIT 1;
```

**Explanation -**

The first part of the code updates the date and here I have used update command and second part of the code is used to display it.



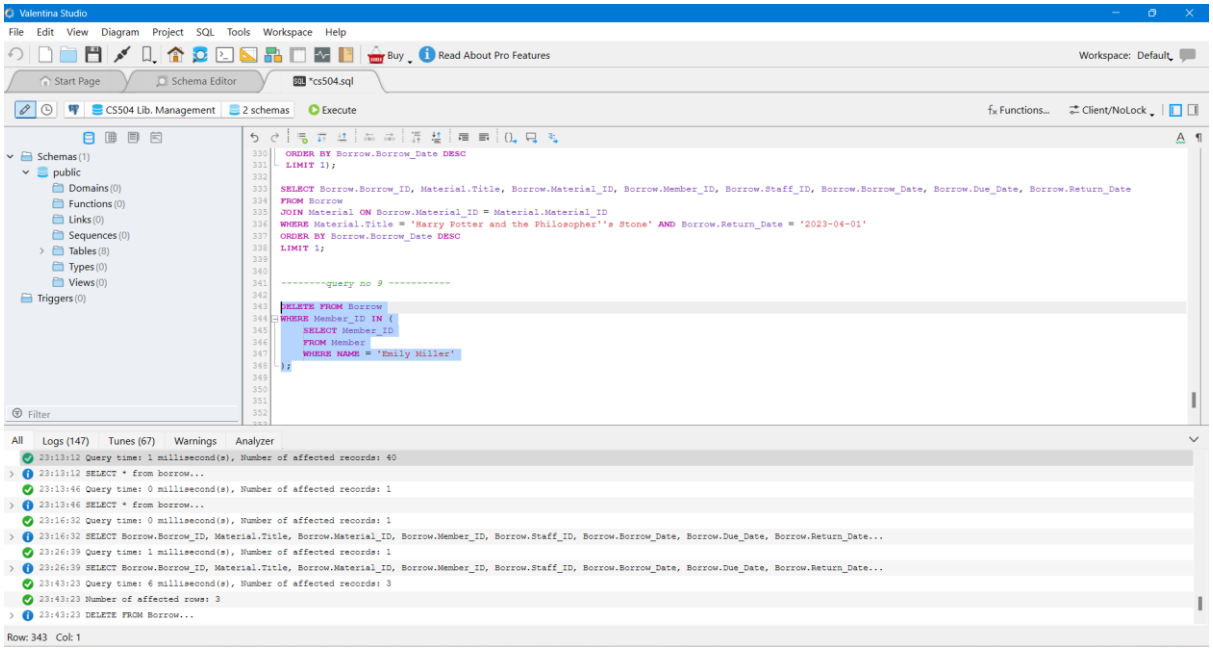

**(9) How do you delete the member Emily Miller and all her related records from the database?**

**Code –**

```
DELETE FROM Borrow
WHERE Member_ID IN (
   SELECT Member_ID
   FROM Member
   WHERE name = 'Emily Miller'
);
```

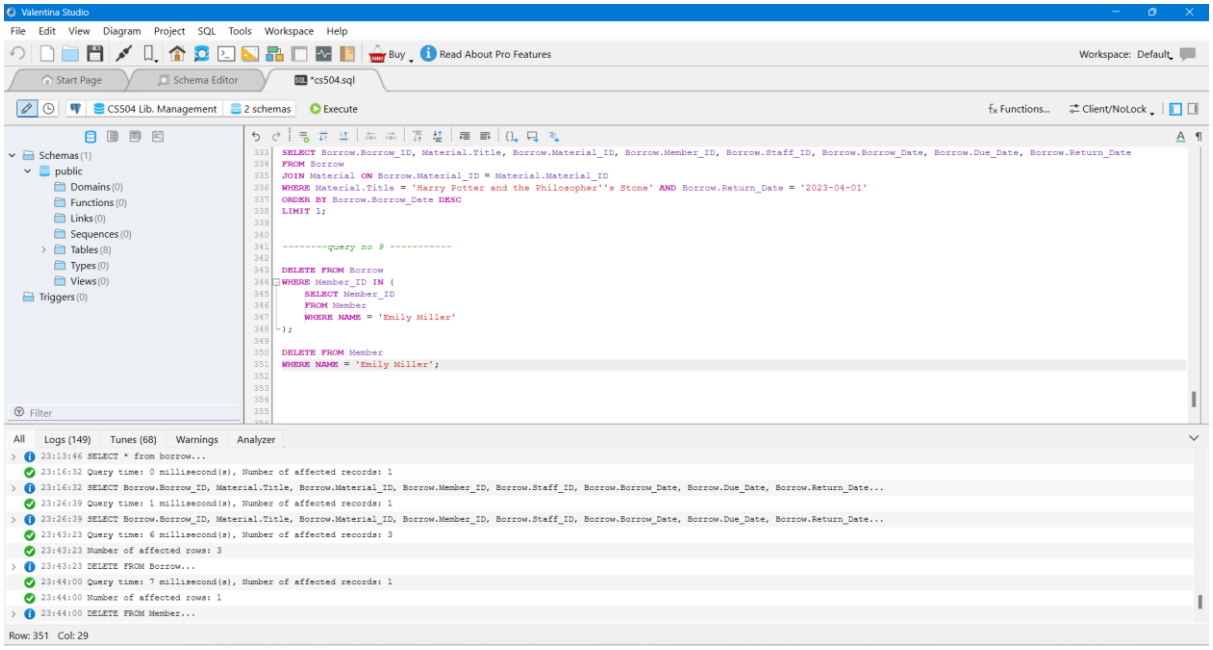
Code –

```
DELETE FROM Member
WHERE name = 'Emily Miller';
```




## Code –

The first part of the code is used to delete the data from the borrow column and the second part of the code basically removes the data from the Member column.



**(10) How do you add the following material to the database?**

**Title: New book**

**Date: 2020-08-01**

**Catalog: E-Books**

**Genre: Mystery & Thriller**

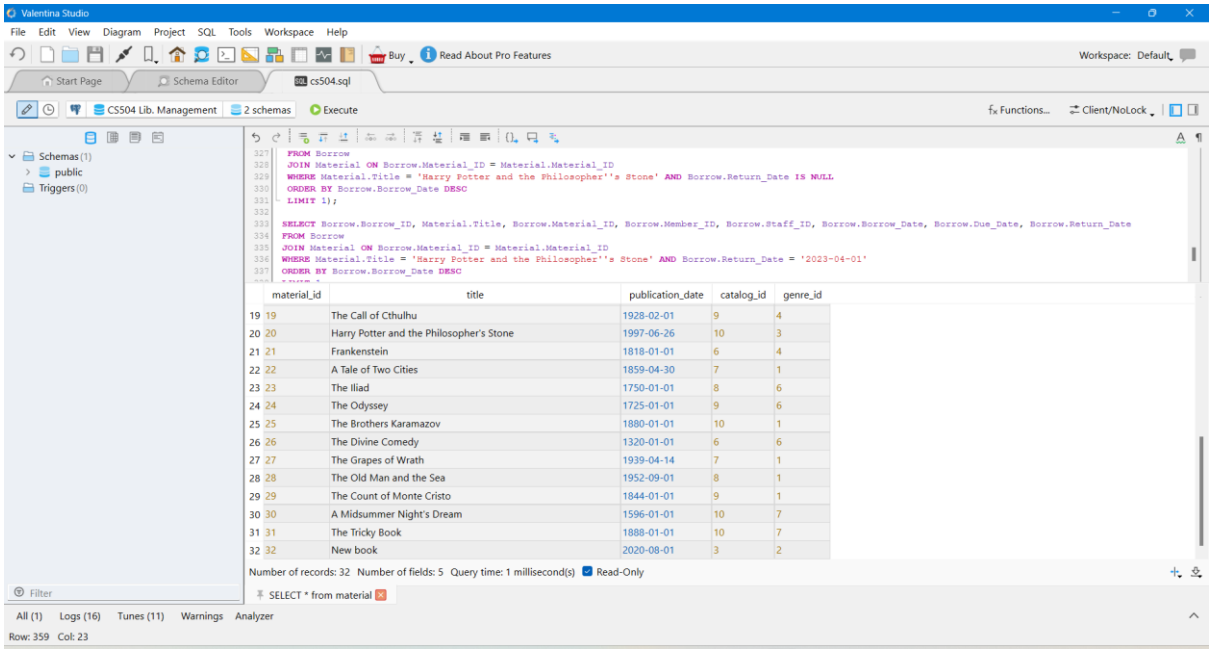
**Author: Lucas Luke**

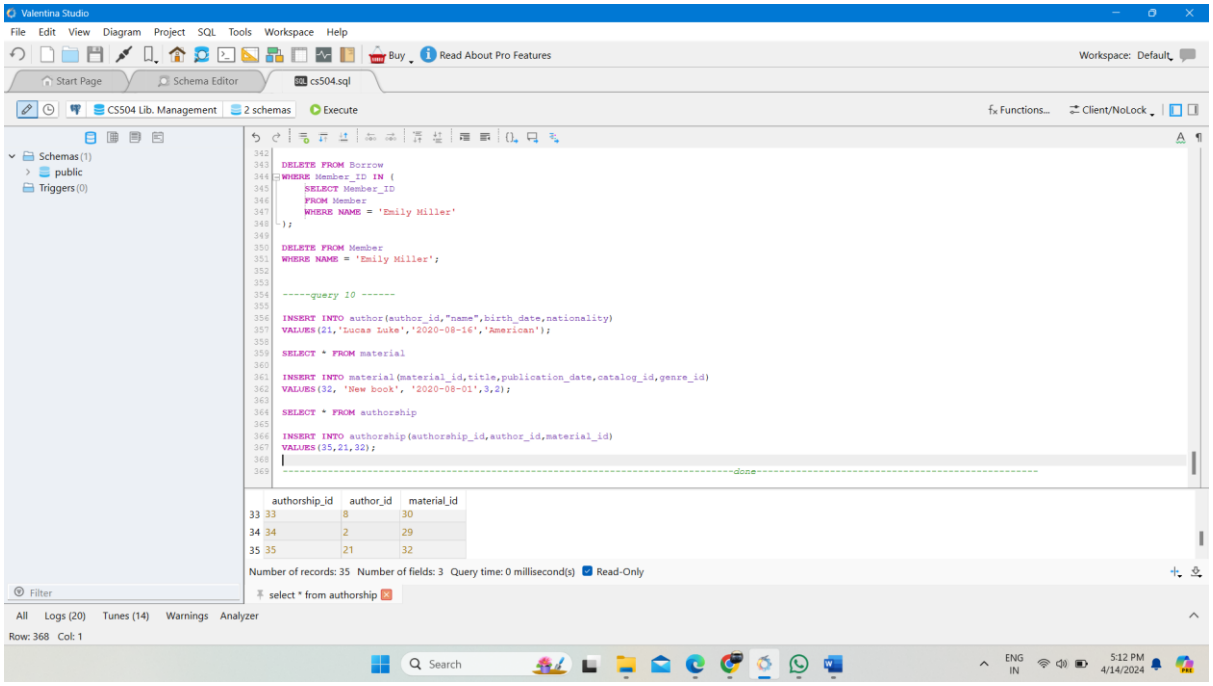
**Code –**

```
SELECT * From Authors

SELECT Genre_ID FROM Genre WHERE Name = 'Mystery & Thriller';

 INSERT INTO author(author_id,"name",birth_date,nationality)
VALUES(21,'Lucas Luke','2020-08-16','American');

INSERT INTO material(material_id,title,publication_date,catalog_id,genre_id)
VALUES(32, 'New book', '2020-08-01',3,2);

SELECT * from material

INSERT INTO authorship(authorship_id,author_id,material_id)
VALUES(35,21,32);
```

**Explanation-**

First I used the query to display new author_id and then I used the query to display the genre_id for genre “Mystery & Thriller” and Material_id from material table then,I first inserted the data into Author table. Then into Material table and then lastly into Authoship table.


Valentina Studio
File Edit View Diagram Project SQL Tools Workspace Help
Buy Read About Pro Features
Workspace: Default
Start Page Schema Editor cs504.sql
CS504 Lib. Management 2 schemas Execute
Functions... Client/NoLock
Schemas (1)
public
Triggers (0)
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return_Date IS NULL
ORDER BY Borrow.Borrow_Date DESC
LIMIT 1);
SELECT Borrow.Borrow_ID, Material.Title, Borrow.Material_ID, Borrow.Member_ID, Borrow.Staff_ID, Borrow.Borrow_Date, Borrow.Due_Date, Borrow.Return_Date
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
WHERE Material.Title = 'Harry Potter and the Philosopher''s Stone' AND Borrow.Return_Date = '2023-04-01'
ORDER BY Borrow.Borrow_Date DESC
material_id title publication_date catalog_id genre_id
19 19 The Call of Cthulhu 1928-02-01 9 4
20 20 Harry Potter and the Philosopher's Stone 1997-06-26 10 3
21 21 Frankenstein 1818-01-01 6 4
22 22 A Tale of Two Cities 1859-04-30 7 1
23 23 The Iliad 1750-01-01 8 6
24 24 The Odyssey 1725-01-01 9 6
25 25 The Brothers Karamazov 1880-01-01 10 1
26 26 The Divine Comedy 1320-01-01 6 6
27 27 The Grapes of Wrath 1939-04-14 7 1
28 28 The Old Man and the Sea 1952-09-01 8 1
29 29 The Count of Monte Cristo 1844-01-01 9 1
30 30 A Midsummer Night's Dream 1596-01-01 10 7
31 31 The Tricky Book 1888-01-01 10 7
32 32 New book 2020-08-01 3 2
Number of records: 32 Number of fields: 5 Query time: 1 millisecond(s) Read-Only
SELECT * from material
All (1) Logs (16) Tunes (11) Warnings Analyzer
Row: 359 Col: 23


Valentina Studio
File Edit View Diagram Project SQL Tools Workspace Help
Buy Read About Pro Features
Workspace: Default
Start Page Schema Editor cs504.sql
CS504 Lib. Management 2 schemas Execute
Functions... Client/NoLock
Schemas (1)
public
Triggers (0)
DELETE FROM Borrow
WHERE Member_ID IN (
SELECT Member_ID
FROM Member
WHERE NAME = 'Emily Miller'
);
DELETE FROM Member
WHERE NAME = 'Emily Miller';
-----query 10 ------
INSERT INTO author(author_id,"name",birth_date,nationality)
VALUES(21,'Lucas Luke','2020-08-16','American');
SELECT * FROM material
INSERT INTO material(material_id,title,publication_date,catalog_id,genre_id)
VALUES(32, 'New book', '2020-08-01',3,2);
SELECT * FROM authorship
INSERT INTO authorship(authorship_id,author_id,material_id)
VALUES(35,21,32);
----------done----------
authorship_id author_id material_id
33 33 8 30
34 34 2 29
35 35 21 32
Number of records: 35 Number of fields: 3 Query time: 0 millisecond(s) Read-Only
select * from authorship
All Logs (20) Tunes (14) Warnings Analyzer
Row: 368 Col: 1
Search
ENG IN
5:12 PM 4/14/2024

## Extended Design:-

### (1) Alert staff about overdue materials on a daily-basis?

For this first I need to check the overdue materials and need to alert the staff about the same here we can divide the query into two ways, part one can be named as creating stored procedure and part will be execution.

**Query for type 1:-**

```
DELIMITER //

CREATE PROCEDURE ReportOverdueItemstostaff()

BEGIN

   SELECT Borrow.Borrow_ID, Borrow.Borrow_Date, Borrow.Due_Date, Borrow.Material_ID, Borrow.Member_ID, Borrow.Staff_ID, Staff.Name AS Staff_Name, Staff.Contact_Info AS Staff_Contact

   FROM Borrow

   JOIN Staff ON Borrow.Staff_ID = Staff.Staff_ID

   WHERE Borrow.Due_Date < CURRENT_DATE AND Borrow.Return_Date IS NULL;

END//

DELIMITER ;
```

**Query for type 2:-**

```
CREATE EVENT CheckOverdueMaterialitems

ON SCHEDULE EVERY 1 DAY

DO

BEGIN

   CALL ReportOverdueItemstostaff();

END;
```

## (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.

For this, I believe we can create a trigger to automatically deactivate the membership, reactivate it .

(1) Part 1 creates a trigger that fires immediately after updating the Borrow table. If a book is returned past its due date, the Overdue_Count for the associated member in the Member table increments. This accumulates the times a member takes late material.

```
ALTER TABLE Member ADD COLUMN Overdue_Count INT DEFAULT 0;

DELIMITER //

CREATE TRIGGER Overdueitems

AFTER UPDATE ON Borrow

FOR EACH ROW

BEGIN

  IF NEW.Due_Date < CURRENT_DATE AND NEW.Return_Date IS NULL THEN

    UPDATE Member SET Overdue_Count = Overdue_Count + 1 WHERE Member_ID = NEW.Member_ID;

  END IF;

END //

DELIMITER ;
```

(2) When executing the check, it looks at overdue counts per member. If the count of overdue payments for one member is greater than or equal to three, the status of membership will be changed to inactive, which effectively deactivates the membership. This function needs to be run on a regular basis for keeping membership statuses up to date.

```
DELIMITER //

CREATE PROCEDURE DeactivateMembershipid()

BEGIN

  UPDATE Member

  SET IsActive = FALSE
```

```
    WHERE Overdue_Count >= 3;
END //
DELIMITER ;
```

(3) Stored procedure, that whenever called with the ID of a specific member resets his/her overdue item count to zero, reactivates the membership, makes possible the borrowing of materials after payment of fines (paid overdue items).

```
DELIMITER //
CREATE PROCEDURE ReactivateMembershipid(member_id INT)
BEGIN
    UPDATE Member
    SET IsActive = TRUE, Overdue_Count = 0
    WHERE Member_ID = member_id;
END //
DELIMITER ;
```

## Conclusion: –

Finally, a relational database management system for the public library is definitely a step forward in the management and access to library resources. With the transition from traditional methodologies to a strong, relational database, the library can expect increased efficiency, fewer errors, and greater user satisfaction. The relational database system, which works on primary entities such as Material, Borrow, Author, Catalog, Genre, Member, and Staff, provides a compact, easy-to-use design that can serve as a base for future development, as shown by the Entity-Relationship Diagram. This project not only streamlines operations but also builds a foundation that could be scaled to meet future improvements. The library can serve its members much better and maintain its processes well without these, which may contain errors in such processes. This setup gives libraries a good level of resilience and the ability to react very quickly to new services.