CS504-INDIVIDUAL PROJECT

-Mounika Jaggampudi

G01359781

Table of Contents

Database Design:

Scope of the project:

The scope of the project is to design and implement a database management system for a public library. The system will manage library resources, including books, magazines, digital media, and other materials, and provide efficient access to member information and facilitate borrowing and tracking of library materials.

The entities and their relationships are:

Material: A library item available for borrowing, such as books, magazines, e-books, and audiobooks.

Catalog: A record of library materials with information on their availability and location.

Genre: A category or type of library materials.

Borrow: The borrowing activity of library materials by members.

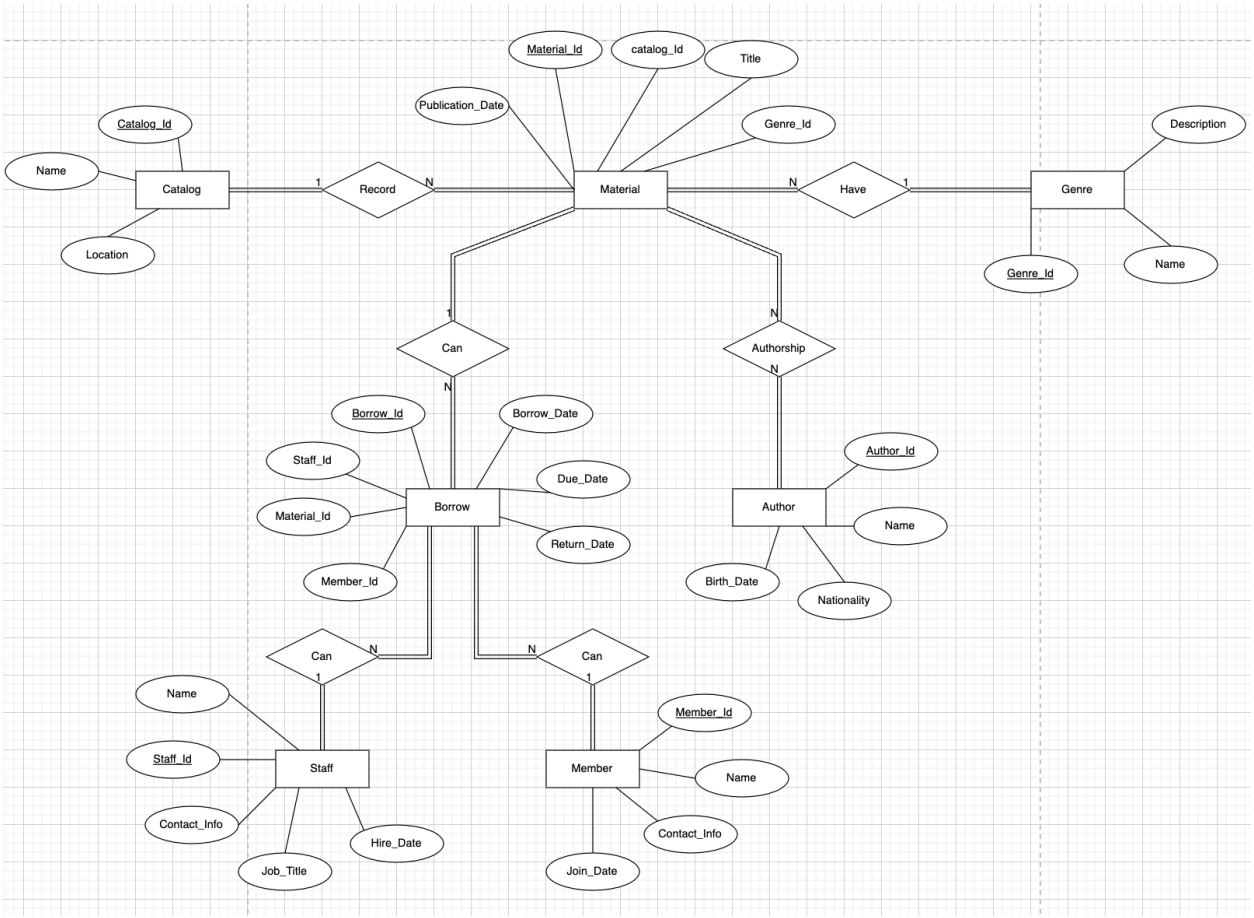
Author: An individual who has created library materials.

Authorship: The relationship between authors and the materials they have created.

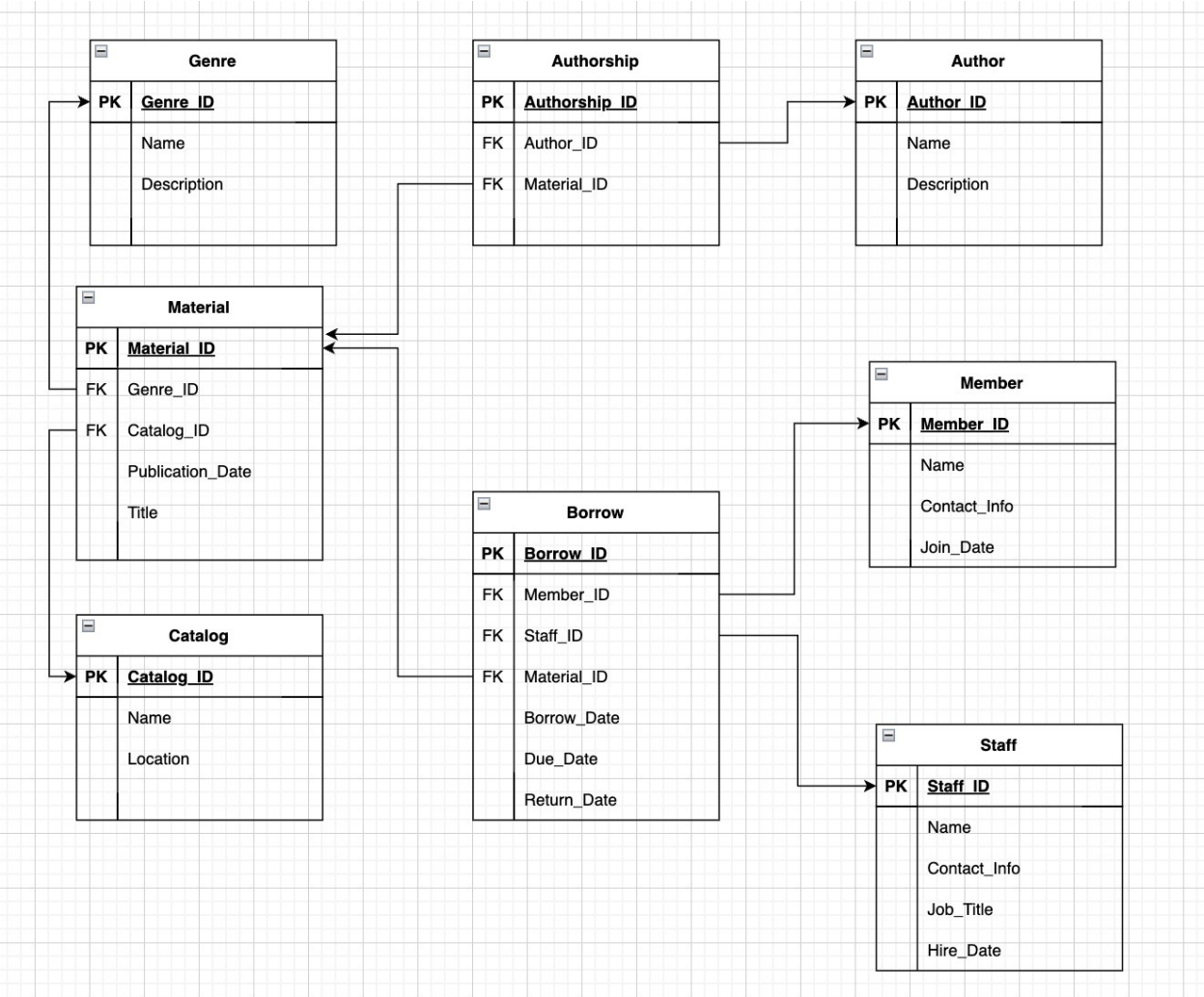
Member: A person who is a member of the library and can borrow and reserve materials.

Staff: A staff member who manages library resources and assists members.

Entity-Relationship (ER) diagram:



Relational Schema Diagram:



Database normalization:

Database normalization is the process of organizing the attributes and tables in a database to minimize redundancy and ensure data integrity. Normalization ensures that the database is free from inconsistencies, update anomalies, and data redundancy, which can affect database performance and accuracy.

The database schema presented in the ER diagram is already normalized. The entities and their relationships are in a third normal form (3NF), which means that each attribute is dependent only on the primary key and there are no transitive dependencies. Therefore, further normalization is not required.

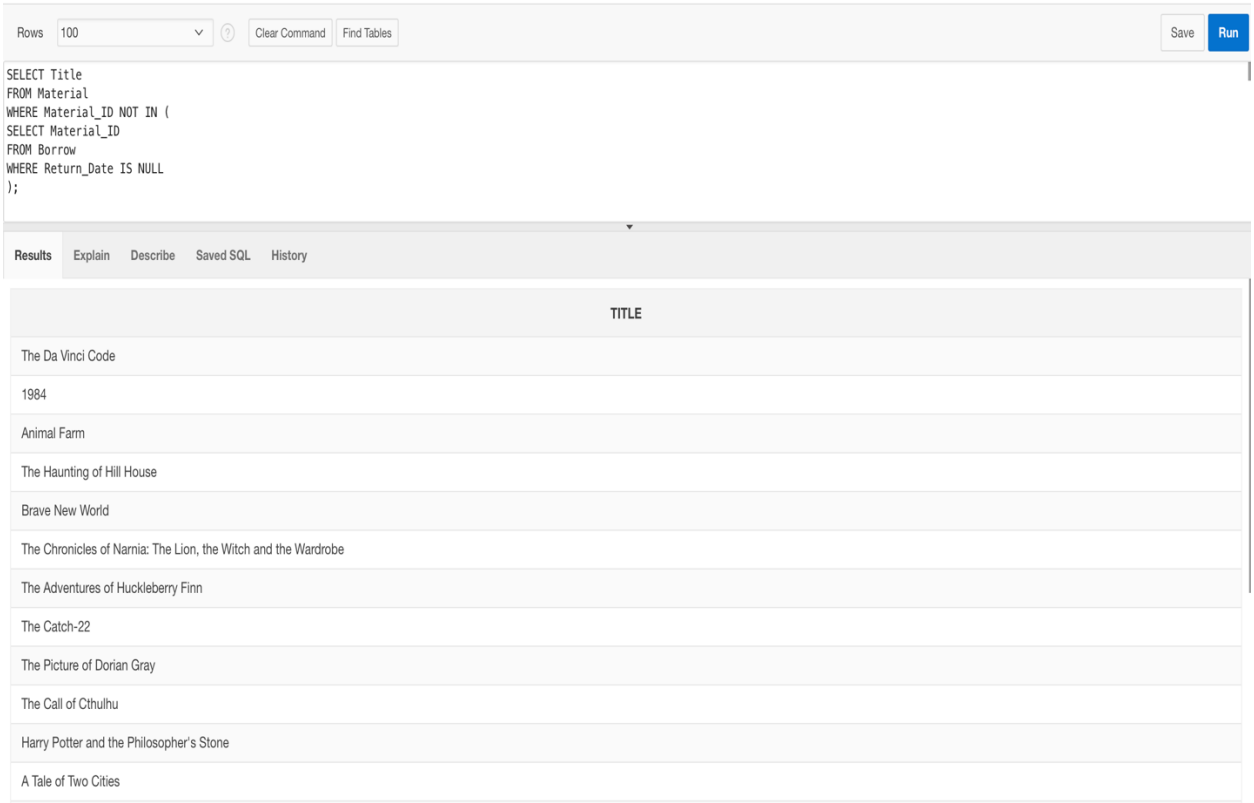
Procedure for Executing Queries:

I used Oracle Database platform for creating public library database and executed all the following queries. Along with every query, there is an image of my execution attached below depicting the result of the query. [The tables that I have created are Material, Catalog, Genre, Author, Member, Staff, Borrow, along with the relationship Authorship. Also inserted the sample data given, I have classified the source codes into two SQL files namely, create_PublicLibrary for creating tables and Insert_PublicLibrary for inserting the sample data.

QUERIES:

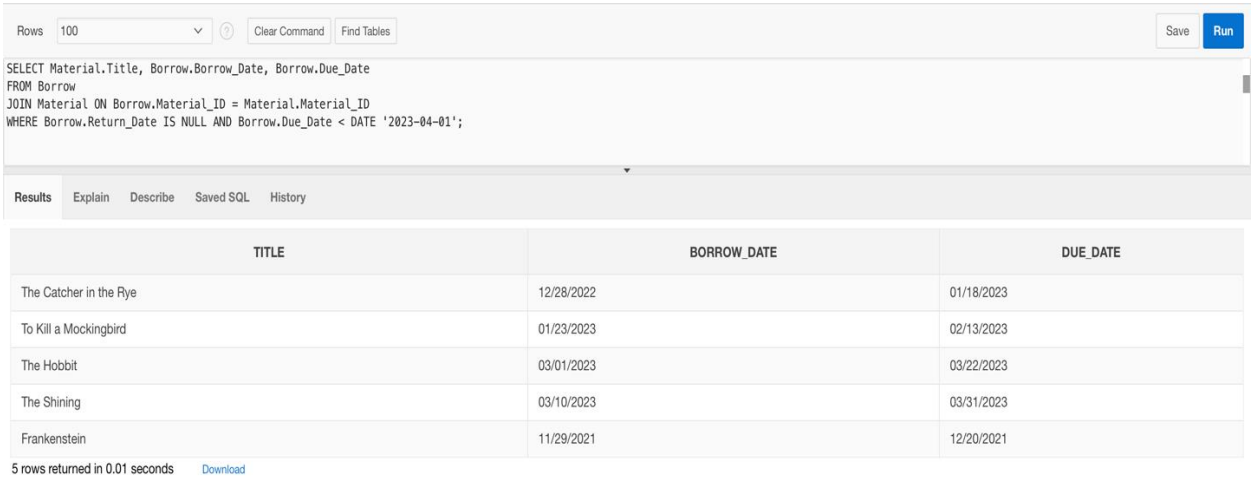
1. Which materials are currently available in the library?

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



2. Which materials are currently overdue?

```
SELECT Material.Title, Borrow.Borrow_Date, Borrow.Due_Date
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
WHERE Borrow.Return_Date IS NULL AND Borrow.Due_Date < DATE '2023-04-01';
```



Rows 100 Clear Command Find Tables Save Run

```
SELECT Material.Title, Borrow.Borrow_Date, Borrow.Due_Date
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
WHERE Borrow.Return_Date IS NULL AND Borrow.Due_Date < DATE '2023-04-01';
```

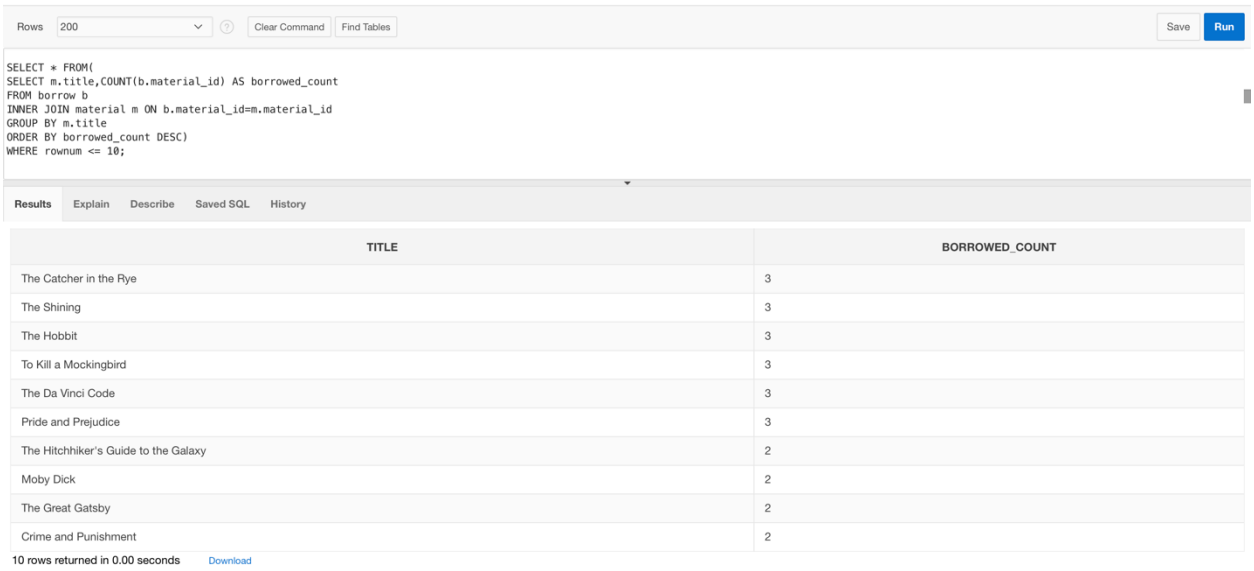
Results Explain Describe Saved SQL History

TITLE	BORROW_DATE	DUE_DATE
The Catcher in the Rye	12/28/2022	01/18/2023
To Kill a Mockingbird	01/23/2023	02/13/2023
The Hobbit	03/01/2023	03/22/2023
The Shining	03/10/2023	03/31/2023
Frankenstein	11/29/2021	12/20/2021

5 rows returned in 0.01 seconds Download

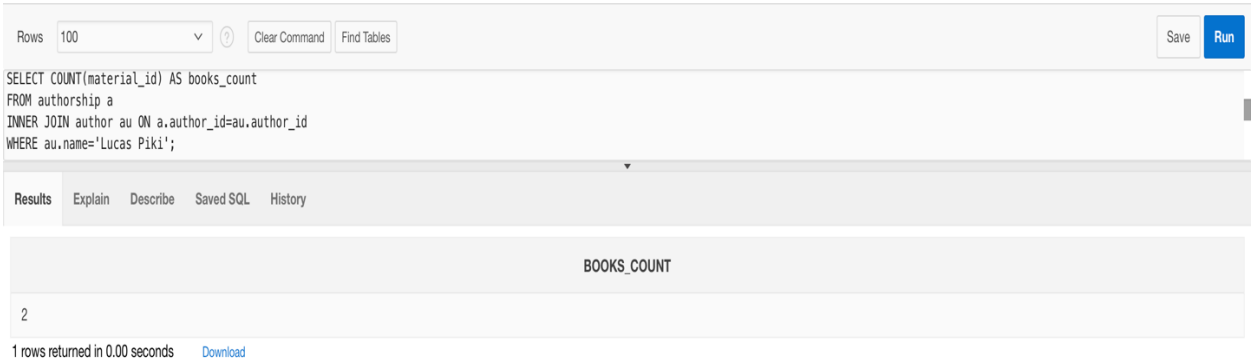
-- 3. What are the top 10 most borrowed materials in the library?

```
SELECT * FROM(
SELECT m.title,COUNT(b.material_id) AS borrowed_count
FROM borrow b
INNER JOIN material m ON b.material_id=m.material_id
GROUP BY m.title
ORDER BY borrowed_count DESC)
WHERE rownum <= 10;
```



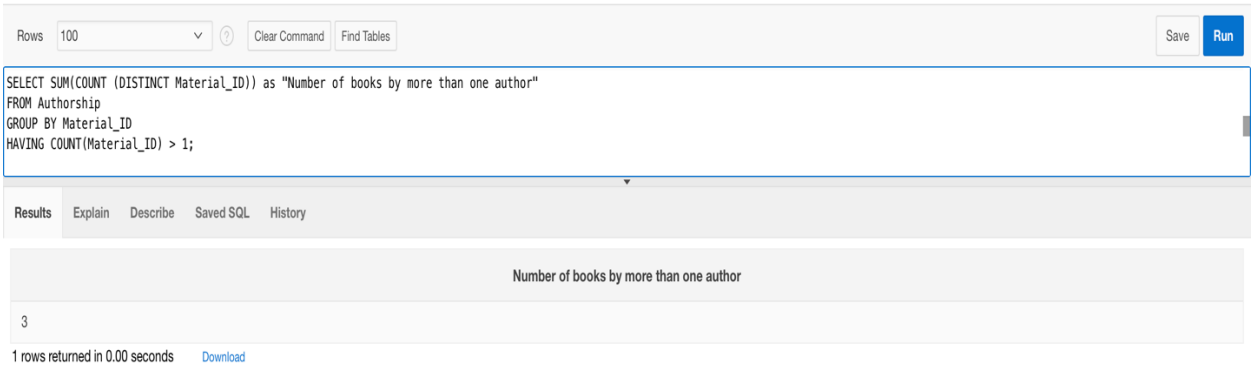
-- 4. How many books has the author Lucas Piki written?

```
SELECT COUNT(material_id) AS books_count
FROM authorship a
INNER JOIN author au ON a.author_id=au.author_id
WHERE au.name='Lucas Piki';
```



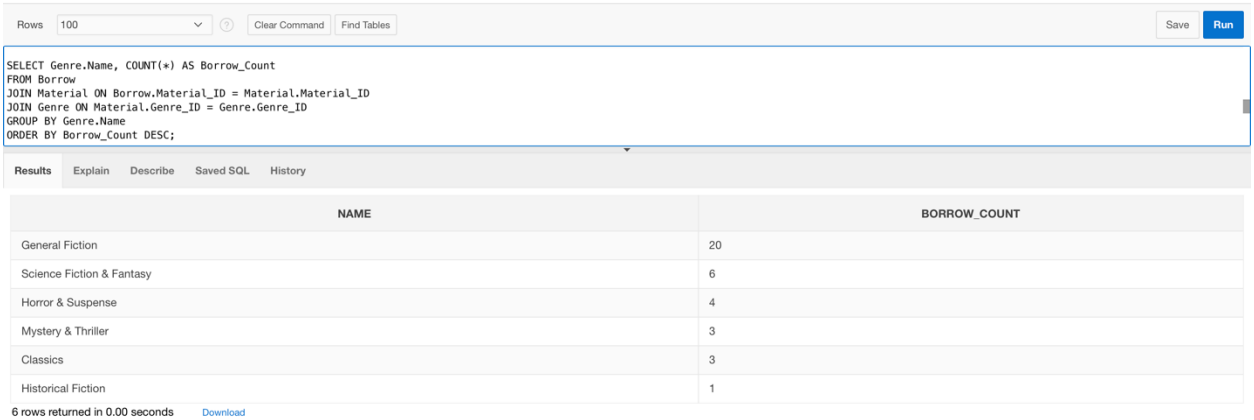
-- 5. How many books were written by two or more authors?

```
SELECT SUM(COUNT (DISTINCT Material_ID)) as "Number of books by more than one author"
FROM Authorship
GROUP BY Material_ID
HAVING COUNT(Material_ID) > 1;
```

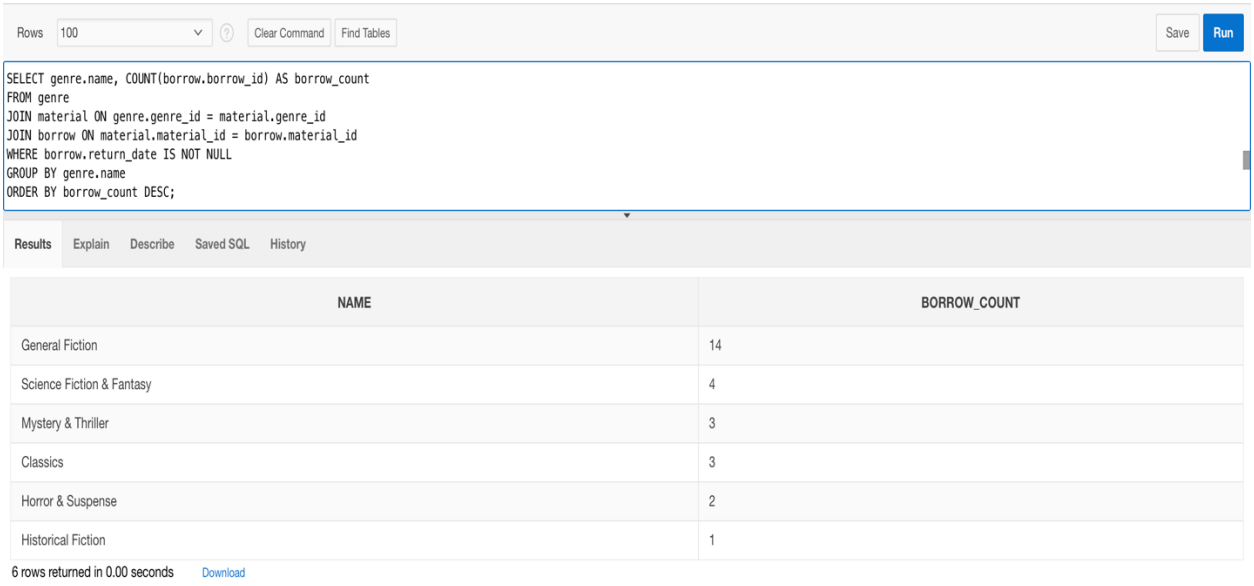


-- 6. What are the most popular genres in the library?

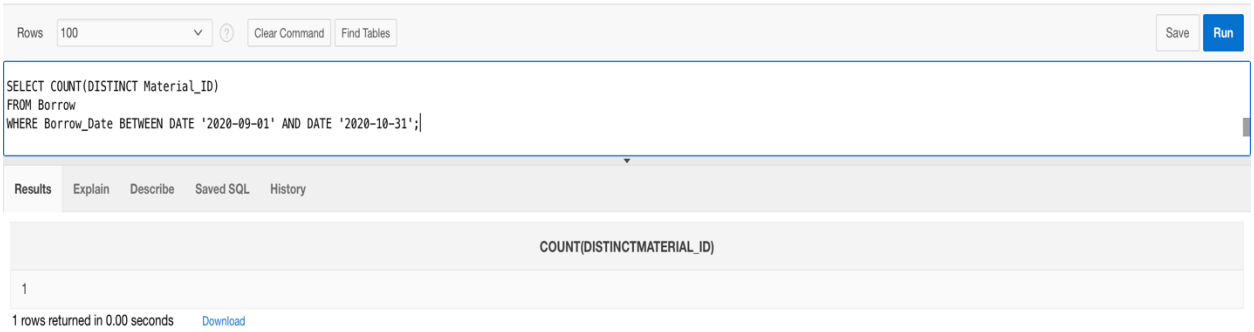
```
SELECT Genre.Name, COUNT(*) AS Borrow_Count
FROM Borrow
JOIN Material ON Borrow.Material_ID = Material.Material_ID
JOIN Genre ON Material.Genre_ID = Genre.Genre_ID
GROUP BY Genre.Genre_ID
ORDER BY Borrow_Count DESC;
```



```
SELECT genre.name, COUNT(borrow.borrow_id) AS borrow_count
FROM genre
JOIN material ON genre.genre_id = material.genre_id
JOIN borrow ON material.material_id = borrow.material_id
WHERE borrow.return_date IS NOT NULL
GROUP BY genre.name
ORDER BY borrow_count DESC;
```




```
-- 7. How many materials have been borrowed from 09/2020-10/2020?
SELECT COUNT(DISTINCT Material_ID)
FROM Borrow
WHERE Borrow_Date BETWEEN '2020-09-01' AND '2020-10-31';
```



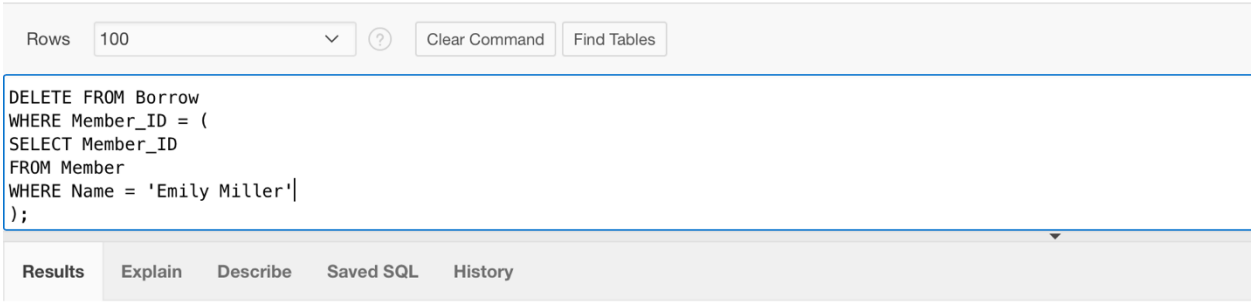
-- 8. How do you update the “Harry Potter and the Philosopher's Stone” when it is returned on

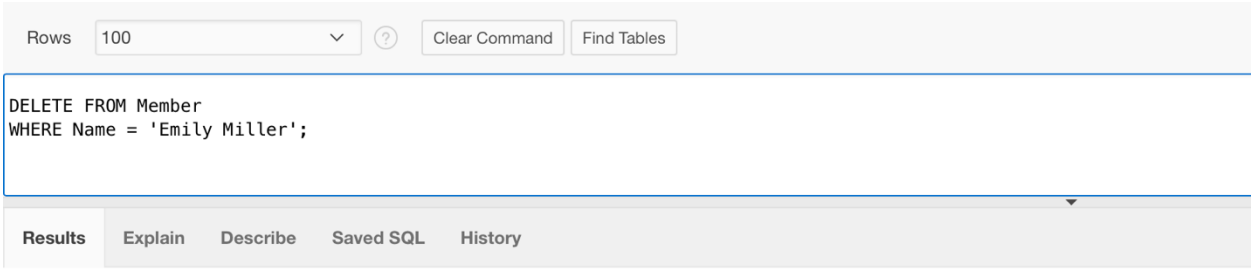
-- 04/01/2023?

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

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

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

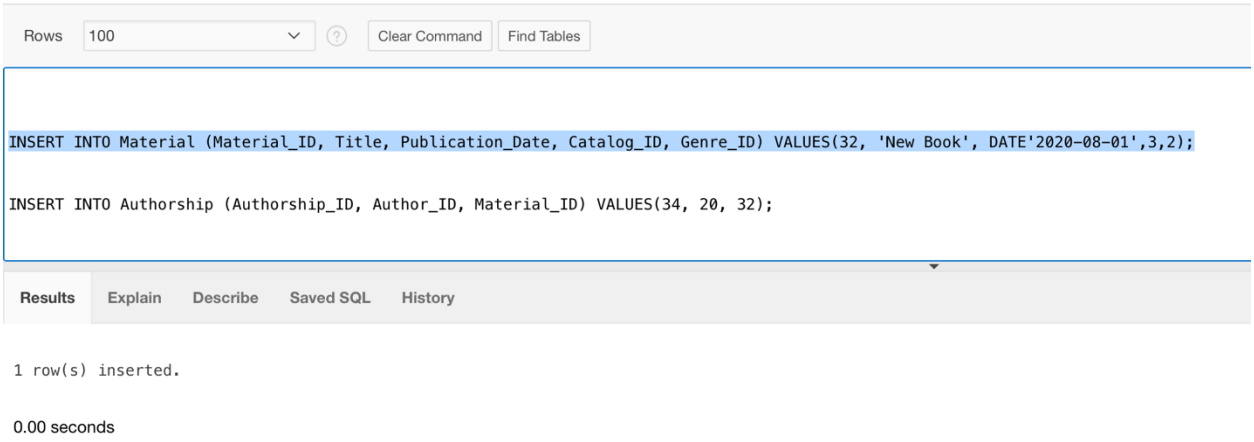


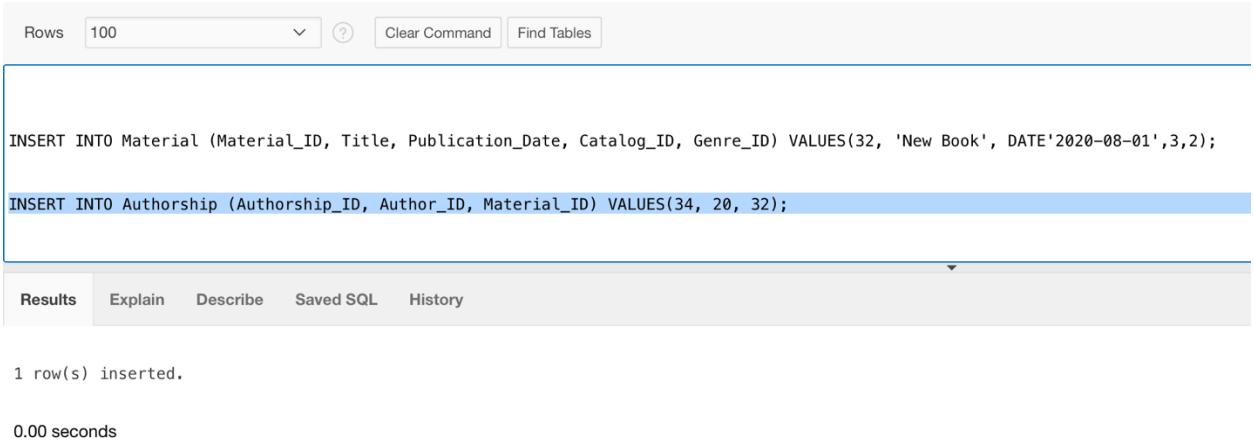


-- 10. How do you add the following material to the database?

INSERT INTO Material (Material_ID, Title, Publication_Date, Catalog_ID, Genre_ID) VALUES(32, 'New Book', DATE'2020-08-01',3,2);

INSERT INTO Authorship (Authorship_ID, Author_ID, Material_ID) VALUES(34, 20, 32);





Design:

1) I can add a script or a stored procedure that runs every day to check for any late materials in the database, extending the current database system to notify personnel about overdue materials daily. To locate any items that still have a member-checked-out status but have a due date that is earlier than today, the script would need to query the database. The script can notify staff

members via email or another preferred mode of communication once it discovers any late items.

2) I can construct a trigger that is invoked when a member checks out a material to automatically deactivate a membership depending on the member's overdue occurrence (>= three times) and revive it once the member pays the overdue cost. If the member has any late occurrences larger than or equal to three, the trigger will need to check that fact. If so, the member's status would be changed to "deactivated." You can change the status to "active" after the member pays the past-due charge.