Name: Nandesh T

UID: 24BAI70121

Course: BE-CSE (AI&ML)

Subject: Database Management Systems

Experiment: Library Management System Implementation

1. Aim of the Session

The aim of this practical is to design and implement a relational database schema for a Library Management System. This involves defining tables with specific constraints, establishing relationships between entities, and managing database security through role-based access control.

2. Objective of the Session

Upon completing this session, the following objectives were achieved:

- Developed table structures using **Primary Keys**, **Foreign Keys**, and **Check Constraints** for data validation.
- Gained proficiency in **DML (Data Manipulation Language)** operations, specifically INSERT, SELECT, UPDATE, and DELETE.
- Implemented **DCL (Data Control Language)** to manage user roles and granular permissions.
- Maintained referential integrity across multiple related tables (`books`, `library_visitors`, and `book_issue`).

3. Practical / Experiment Steps

The implementation was carried out through the following tasks:

1. **Schema Definition:** Created the base tables for `books` and `library_visitors` with specific constraints such as NOT NULL, UNIQUE, and CHECK (e.g., ensuring visitor age is 18+).
2. **Relational Setup:** Created the `book_issue` table to act as a transaction bridge, linking books and visitors via Foreign Keys.
3. **Data Population:** Populated the tables with initial records to test the schema's validity.

4. **Operational Testing:** Performed updates on user information and attempted deletion of records to observe constraint behavior.
5. **Security Administration:** Created a `librarian` role with login credentials and configured its access levels using `GRANT` and `REVOKE` commands.

4. Procedure of the Practical

The following steps were followed during the execution:

1. **System Initialization:** Logged into the database environment and established a connection to the server.
2. **Database Creation:** Initialized a new database to house the library management system.
3. **Executing Table Scripts:** Ran the `CREATE TABLE` commands in a specific sequence (creating parent tables before dependent transaction tables).
4. **Data Entry:** Executed `INSERT` statements to add sample books and visitor profiles.
5. **Query Verification:** Used `SELECT` queries to verify that the data was correctly stored and consistent across tables.
6. **Data Modification:** Tested the `UPDATE` and `DELETE` commands to ensure the system handles changes as intended.
7. **Role Configuration:** Defined the `librarian` role and assigned specific table privileges.
8. **Security Verification:** Tested and then revoked permissions to confirm the effectiveness of the security policy.
9. **Record Maintenance:** Saved the SQL script and took screenshots of the execution results.

5. I/O Analysis (Input / Output Analysis)

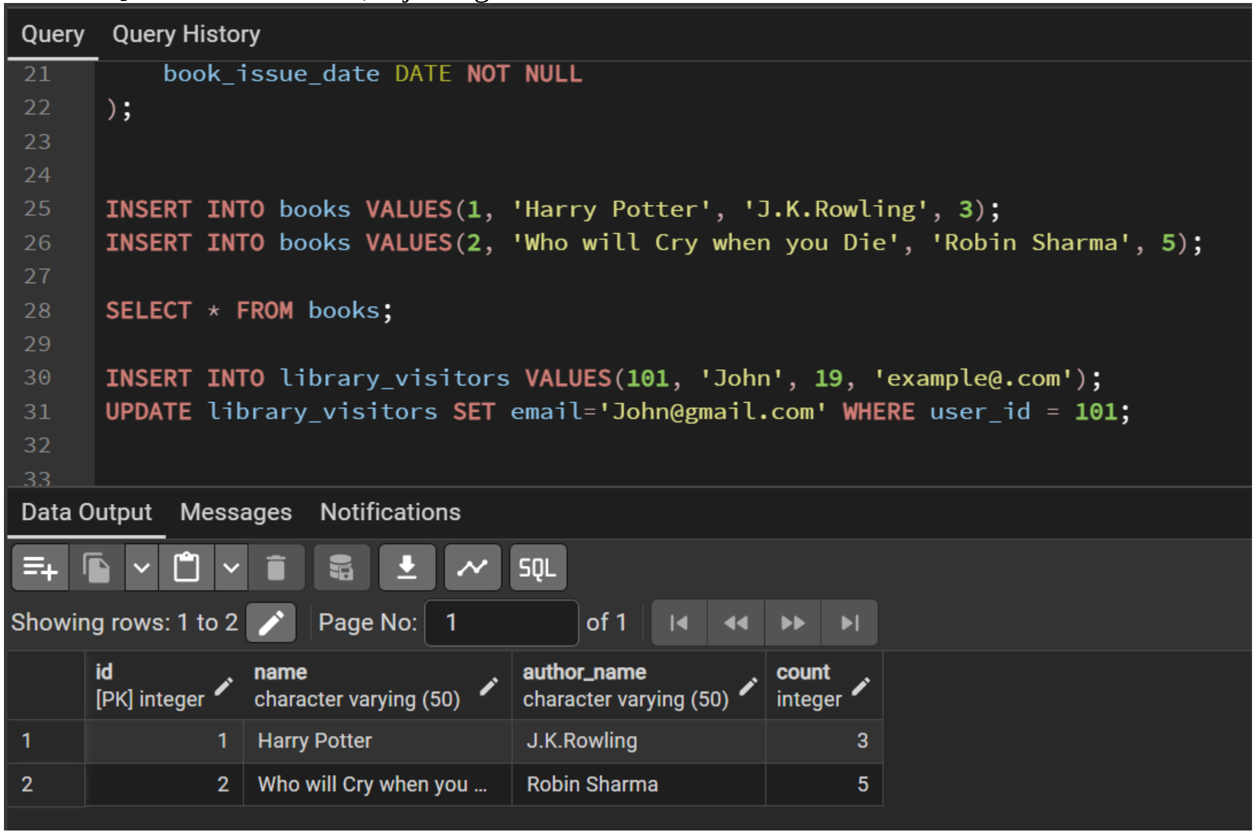
Input Queries

SQL

```
CREATE TABLE books(
    id INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    author_name VARCHAR(50) NOT NULL,
    count INT CHECK(count>0)
);

CREATE TABLE library_visitors(
    user_id INT PRIMARY KEY,
    user_name VARCHAR(20) NOT NULL,
    age INT CHECK(age>=18) NOT NULL,
    email VARCHAR(40) UNIQUE NOT NULL
);

CREATE TABLE book_issue(
    book_issue_id INT PRIMARY KEY,
    book_id INT NOT NULL,
    user_id INT NOT NULL,
    FOREIGN KEY (book_id) REFERENCES books(id),
    FOREIGN KEY (user_id) REFERENCES library_visitors(user_id),
    book_issue_date DATE NOT NULL
);

INSERT INTO books VALUES(1, 'Harry Potter', 'J.K.Rowling', 3);

INSERT INTO library_visitors VALUES(101, 'John', 19, 'example@.com');

UPDATE library_visitors SET email='John@gmail.com' WHERE user_id = 101;

CREATE ROLE librarian WITH LOGIN PASSWORD 'PASSWORD';

GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian;
```

Output Details

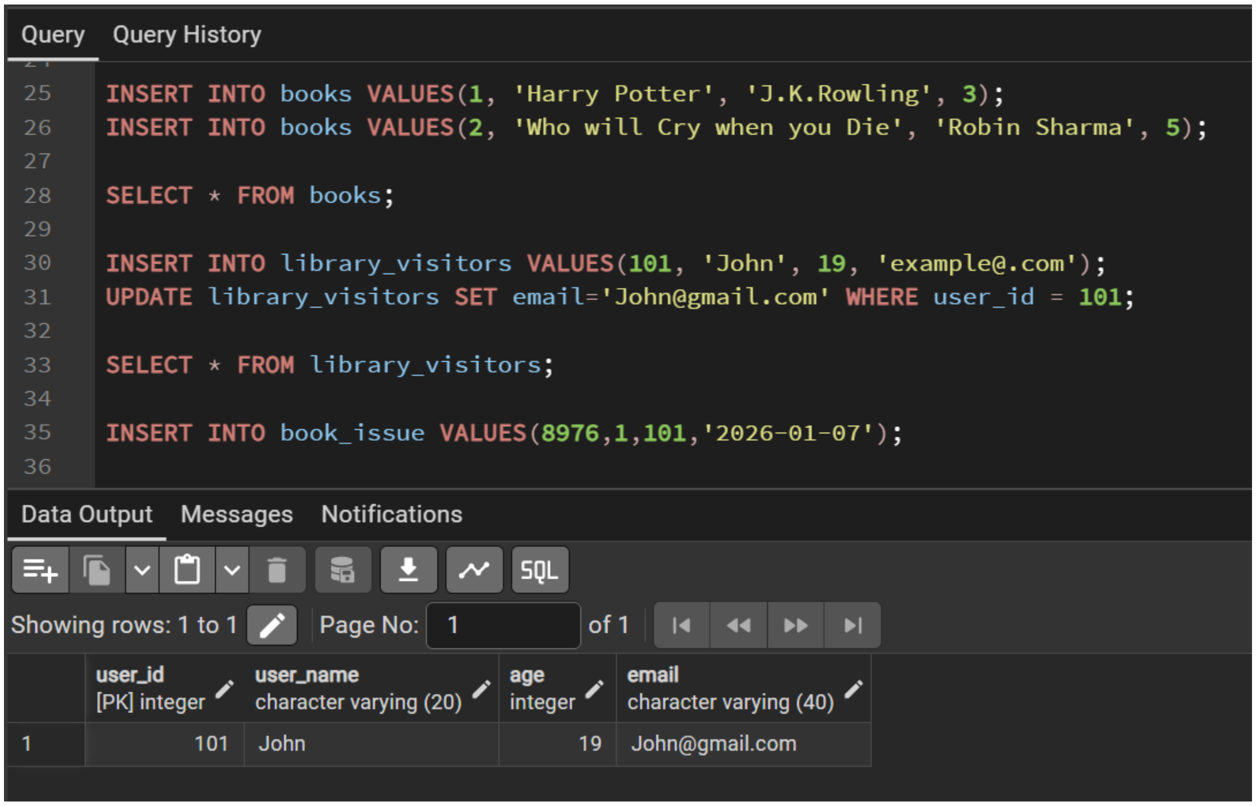
- **Schema Success:** All tables were created successfully. The system correctly enforced the `CHECK(age>=18)` constraint, rejecting invalid entries.



```
    book_issue_date DATE NOT NULL
);


INSERT INTO books VALUES(1, 'Harry Potter', 'J.K.Rowling', 3);
INSERT INTO books VALUES(2, 'Who will Cry when you Die', 'Robin Sharma', 5);

SELECT * FROM books;

INSERT INTO library_visitors VALUES(101, 'John', 19, 'example@.com');
UPDATE library_visitors SET email='John@gmail.com' WHERE user_id = 101;
```

	id [PK] integer	name character varying (50)	author_name character varying (50)	count integer
1	1	Harry Potter	J.K.Rowling	3
2	2	Who will Cry when you ...	Robin Sharma	5

- **DML Results:** The `UPDATE` query correctly modified the email field for user 101, and `SELECT` queries displayed the current state of all tables accurately.



Query | Query History

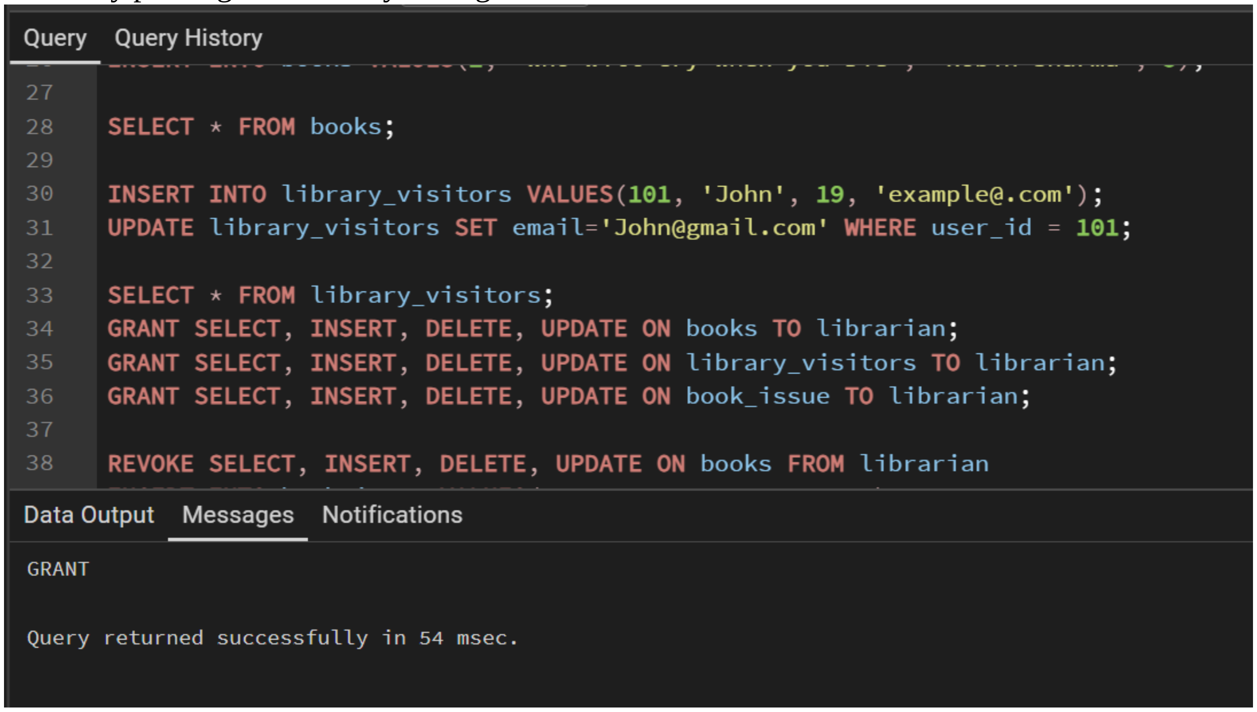
```
INSERT INTO books VALUES(1, 'Harry Potter', 'J.K.Rowling', 3);
INSERT INTO books VALUES(2, 'Who will Cry when you Die', 'Robin Sharma', 5);

SELECT * FROM books;

INSERT INTO library_visitors VALUES(101, 'John', 19, 'example@.com');
UPDATE library_visitors SET email='John@gmail.com' WHERE user_id = 101;

SELECT * FROM library_visitors;

INSERT INTO book_issue VALUES(8976,1,101,'2026-01-07');
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 Page No: 1 of 1

	user_id [PK] integer	user_name character varying (20)	age integer	email character varying (40)
1	101	John	19	John@gmail.com

- **DCL Verification:** The `librarian` role was successfully created and assigned the necessary privileges for library management tasks.



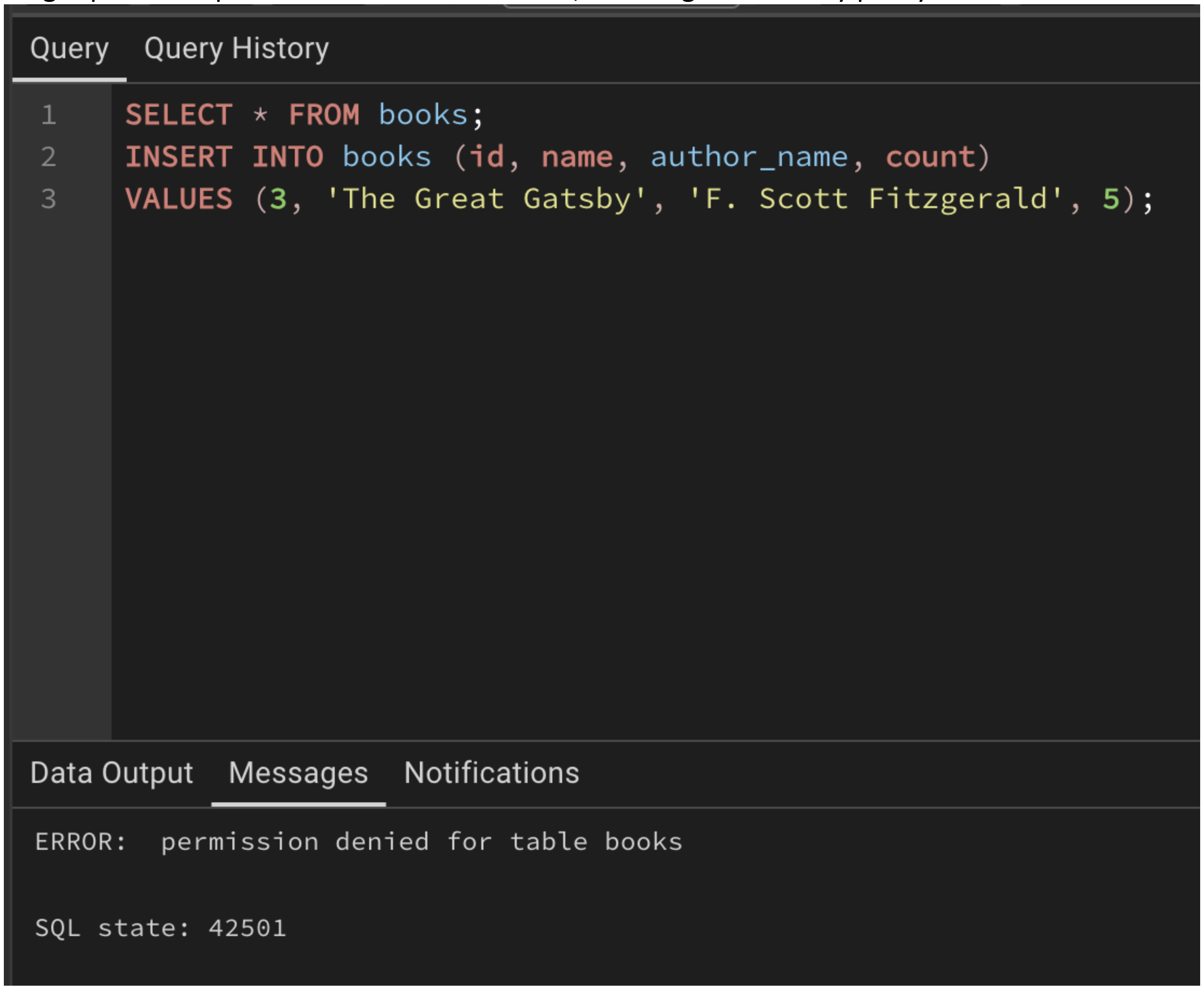
Query | Query History

```
SELECT * FROM books;

INSERT INTO library_visitors VALUES(101, 'John', 19, 'example@.com');
UPDATE library_visitors SET email='John@gmail.com' WHERE user_id = 101;

SELECT * FROM library_visitors;
GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian;
GRANT SELECT, INSERT, DELETE, UPDATE ON library_visitors TO librarian;
GRANT SELECT, INSERT, DELETE, UPDATE ON book_issue TO librarian;

REVOKE SELECT, INSERT, DELETE, UPDATE ON books FROM librarian
```

Data Output | Messages | Notifications

```
GRANT

Query returned successfully in 54 msec.
```

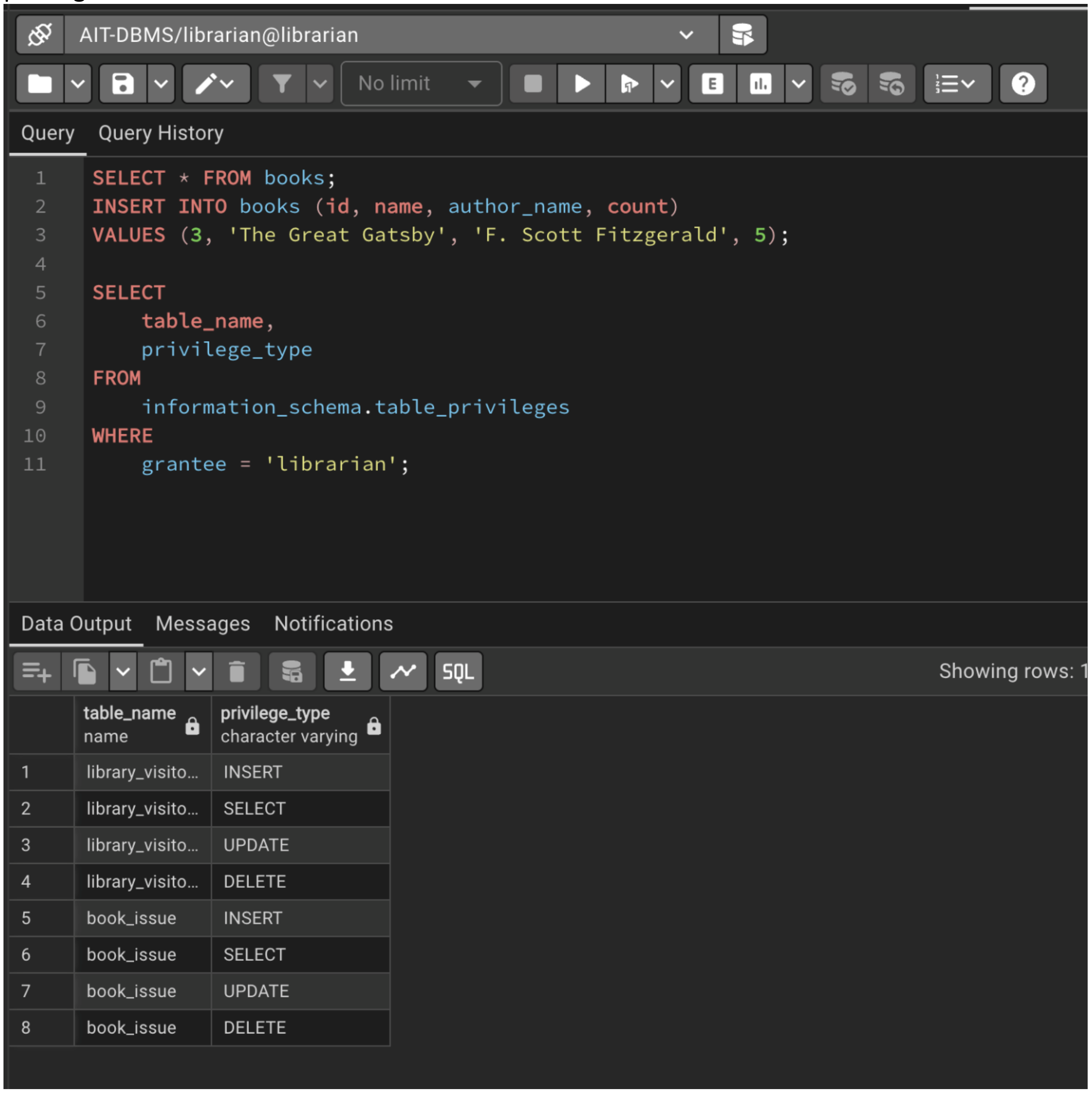
- **Validation:** Testing confirmed that after the `REVOKE` command, the `librarian` could no longer perform operations on the `books` table, ensuring the security policy is functional.



```sql
SELECT * FROM books;
INSERT INTO books (id, name, author_name, count)
VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 5);
```

```
ERROR:  permission denied for table books

SQL state: 42501
```

- We also confirmed the permissions of the role “librarian” by checking the table privileges.



```sql
SELECT * FROM books;
INSERT INTO books (id, name, author_name, count)
VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 5);

SELECT
    table_name,
    privilege_type
FROM
    information_schema.table_privileges
WHERE
    grantee = 'librarian';
```

	table_name name	privilege_type character varying
1	library_visito...	INSERT
2	library_visito...	SELECT
3	library_visito...	UPDATE
4	library_visito...	DELETE
5	book_issue	INSERT
6	book_issue	SELECT
7	book_issue	UPDATE
8	book_issue	DELETE

6. Learning Outcome

This practical session provided significant insights into:

- **Structural Logic:** Understanding how Foreign Keys and Check Constraints maintain high data quality and prevent logical errors.
- **Security Implementation:** Learning to manage database security through roles rather than individual user permissions.
- **Practical Application:** Applying SQL fundamentals to a real-world scenario (Library Management), demonstrating how relational databases handle complex interactions between entities.