Student Name: Shreyansh Gupta UID: 24BAI70658
Branch: AIT-CSE (AIML) Section/Group: 24AIT_KRG-/G2
Semester: 4
Subject Name: Database Management System Subject Code: 24CSH-298

Experiment

Experiment 1.1: Design and implementation of a Library Management System using PostgreSQL with DDL, DML and DCL commands.

Aim

The aim of this experiment is to design and implement a Library Management System database using PostgreSQL. The database is created using proper tables, primary keys, foreign keys and constraints. DML operations are performed and database security is implemented using roles and privileges.

Objective

The objective of this experiment is to gain practical knowledge of DDL, DML and DCL commands in PostgreSQL. It also helps in understanding how to create roles, grant permissions and revoke permissions to secure the database using role based access control.

Practical / Experiment Steps

1. Design the database structure for the Library Management System.
2. Create tables for books, members and issue records using DDL commands.
3. Apply primary keys, foreign keys and constraints to maintain data integrity.
4. Insert sample records into the tables using DML commands.
5. Update and delete records as required.
6. Create a database role named Librarian.
7. Grant required permissions like SELECT, INSERT and DELETE to the Librarian role.
8. Revoke permissions when needed to ensure database security.

Procedure of the Experiment

1. Start the system and log in to the computer.

2. Open pgAdmin and connect to PostgreSQL server.

3. Create a new database for the Library Management System.

4. Create tables such as Books, Members and Issue_Records using CREATE TABLE command.

5. Define primary keys and foreign keys while creating the tables.

6. Insert records into tables using INSERT command.

7. Update existing data using UPDATE command.

8. Delete unwanted records using DELETE command.

9. Create a role named Librarian with password using CREATE ROLE command.

10. Grant SELECT, INSERT and DELETE permissions to the Librarian role.

11. Revoke permissions using REVOKE command when required.

12. Execute all queries and verify the output.

CODE :

1. **ADMIN**

```
CREATE TABLE BOOKS(

ID INT PRIMARY KEY,
NAME VARCHAR(20) NOT NULL,
AUTHOR_NAME VARCHAR(20) NOT NULL,
COUNT INT
)
INSERT INTO BOOKS VALUES(1,'HARRY POTTER','JK Rowling',2);
SELECT * FROM BOOKS;
UPDATE BOOKS
SET COUNT = 3
WHERE ID=1
CREATE TABLE LIBRARY_VISITOR_USER(
USER_ID INT PRIMARY KEY,
USER_NAME VARCHAR(20),
AGE INT CHECK(AGE>=17) NOT NULL,
EMAIL VARCHAR(20) UNIQUE NOT NULL
)
INSERT INTO LIBRARY_VISITOR_USER(USER_ID,USER_NAME,AGE,EMAIL)
VALUES (101,'AKASH SHARMA',19, 'AKSH@GMAIL.COM')
SELECT * FROM LIBRARY_VISITOR_USER
UPDATE LIBRARY_VISITOR_USER
SET EMAIL='AK@GMAIL.COM'
SELECT * FROM LIBRARY_VISITOR_USER
CREATE TABLE BOOK_ISSUE(
BOOK_ISSUE_ID INT PRIMARY KEY,
BOOK_ID INT REFERENCES  BOOKS(ID) NOT NULL,
USER_ID INT REFERENCES  LIBRARY_VISITOR_USER(USER_ID) NOT NULL,
BOOK_ISSUE DATE NOT NULL
)
INSERT INTO BOOK_ISSUE VALUES(5552,1,101,'2026-01-09')
SELECT * FROM BOOK_ISSUE
CREATE ROLE LIBRARIAN
WITH LOGIN PASSWORD '123123'
GRANT SELECT ,INSERT, DELETE,UPDATE ON BOOKS TO LIBRARIAN
GRANT SELECT ,INSERT, DELETE,UPDATE ON BOOK_ISSUE TO LIBRARIAN
GRANT SELECT ,INSERT, DELETE,UPDATE ON LIBRARY_VISITOR_USER TO LIBRARIAN
REVOKE SELECT ,INSERT, DELETE,UPDATE ON BOOKS FROM LIBRARIANLIBRARIAN
```

2.LIBRARIAN

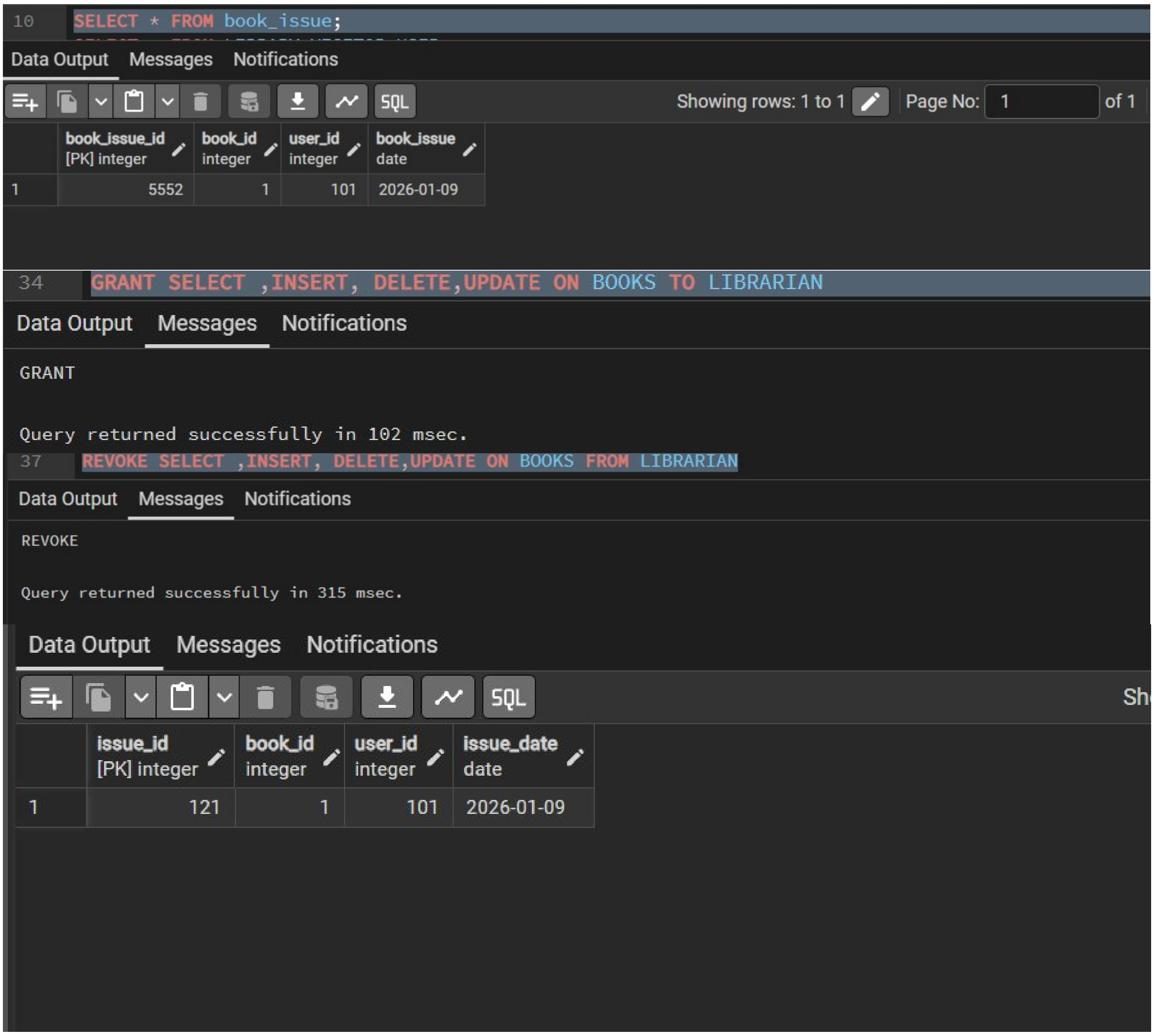
```
SELECT * FROM books;

select * from book_issue;

select * from LIBRARY_VISITOR_USER;


INSERT INTO books VALUES(150,'HARRY POTTER 2','JK ROWLING',4);


DELETE FROM books

WHERE ID=150;


SELECT * FROM book_issue;

SELECT * FROM LIBRARY_VISITOR_USER;
```

Learning Outcomes:

1. Understood the basics of **relational database design** using tables, keys, and relationships.
2. Learned to apply **primary key and foreign key constraints** to maintain data integrity.
3. Gained hands-on experience with **INSERT, UPDATE, and DELETE** operations safely.
4. Understood how **roles and privileges** control access to database objects.
5. Learned to use **GRANT and REVOKE** for implementing **read-only users**.

6. Practiced **ALTER TABLE and DROP TABLE** for managing database changes.

SCREENSHOTS



```
SELECT * FROM book_issue;
```

	book_issue_id [PK] integer	book_id integer	user_id integer	book_issue date
1	5552	1	101	2026-01-09

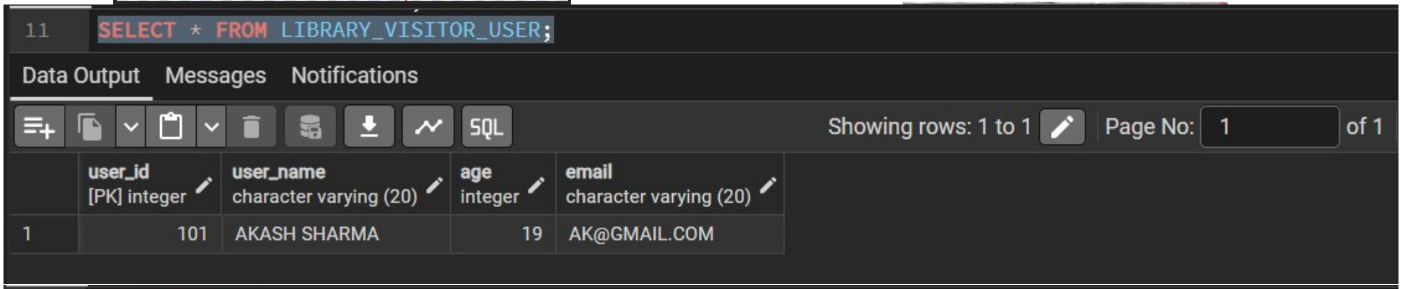
```
GRANT SELECT ,INSERT, DELETE,UPDATE ON BOOKS TO LIBRARIAN
```

```
GRANT

Query returned successfully in 102 msec.
```

```
REVOKE SELECT ,INSERT, DELETE,UPDATE ON BOOKS FROM LIBRARIAN
```

```
REVOKE

Query returned successfully in 315 msec.
```

	issue_id [PK] integer	book_id integer	user_id integer	issue_date date
1	121	1	101	2026-01-09

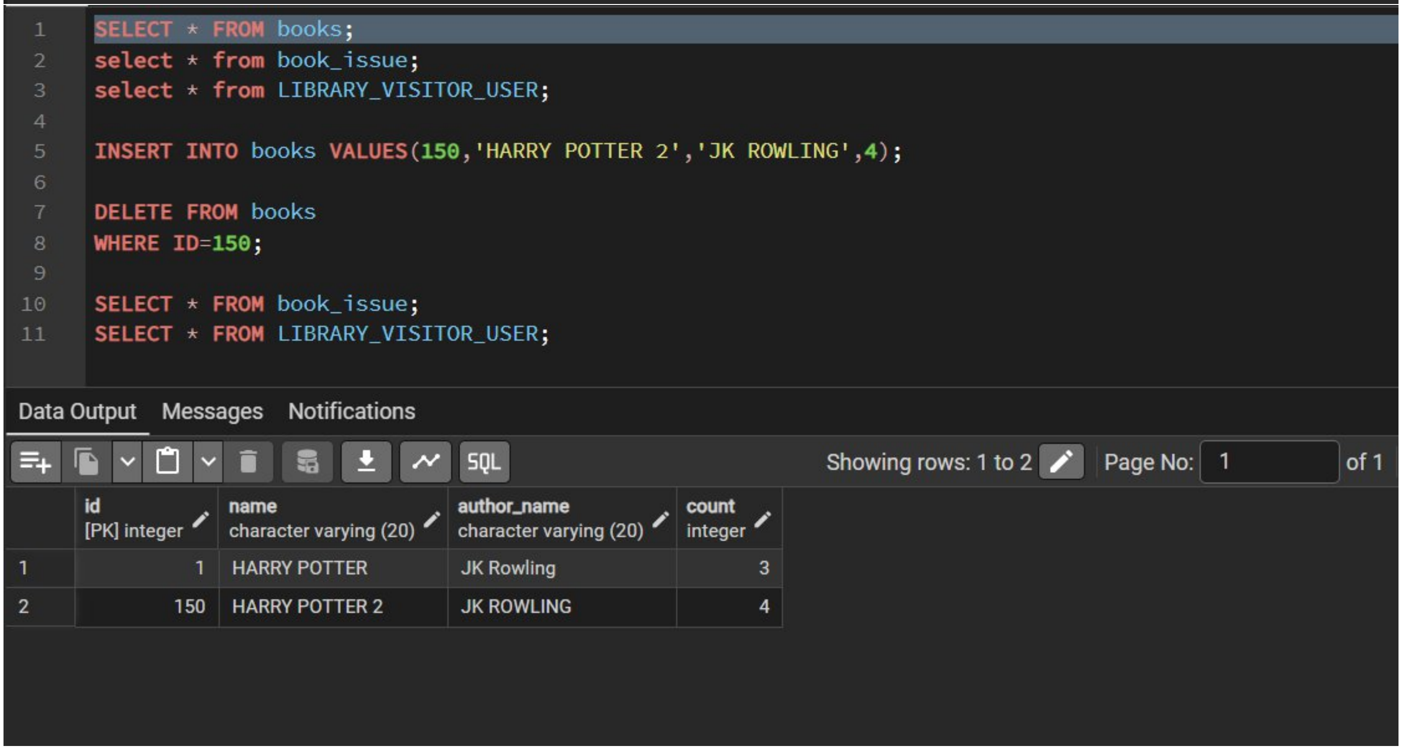


```
SELECT * FROM LIBRARY_VISITOR_USER;
```

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 (20)
1	101	AKASH SHARMA	19	AK@GMAIL.COM



```
SELECT * FROM books;
select * from book_issue;
select * from LIBRARY_VISITOR_USER;

INSERT INTO books VALUES(150,'HARRY POTTER 2','JK ROWLING',4);

DELETE FROM books
WHERE ID=150;

SELECT * FROM book_issue;
SELECT * FROM LIBRARY_VISITOR_USER;
```

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	id [PK] integer	name character varying (20)	author_name character varying (20)	count integer
1	1	HARRY POTTER	JK Rowling	3
2	150	HARRY POTTER 2	JK ROWLING	4