Student Name: Harshit Kumawat **UID: 24BAI70025**
Branch: B.E CSE(A.I & M.L) **Section/Group: 24 AIT-KRG-G1**
Semester: 4TH Sem **Date of Performance:09/01/2026**
Subject Name: DBMS **Subject Code:**

WORKSHEET 1

AIM: To design and implement a Library Management System using SQL by creating tables with constraints, performing data manipulation operations, and managing user roles and privileges.

S/W Requirement: PostgreSQL (pgAdmin 4)/ Oracle

OBJECTIVES:

To create tables using `PRIMARY KEY`, `FOREIGN KEY`, `NOT NULL`, `UNIQUE`, and `CHECK` constraints.

To insert, update, delete, and retrieve records using SQL DML commands.

To maintain referential integrity between tables.

To create a database role and manage access permissions using `GRANT` and `REVOKE`.

PROCEDURE:

1. **Environment Setup**
   - Launch the PostgreSQL administration tool (pgAdmin 4).
   - Create a new database instance for the Library Management System.
2. **Database Design**
   - Execute the `CREATE TABLE`statements for the `books`, `library_visitors`, and `book_issue` tables.
   - Verify that the following constraints are successfully implemented in the respective tables:
     - Primary Keys: `id` in `books`, `user_id` in `library_visitors`, and `book_issue_id`in `book_issue`.
     - Foreign Keys: `book_id` and `user_id` in `book_issue`referencing `books` and `library_visitors`, respectively.
     - NOT NULL: On essential columns like `name`, `author_name`, `user_name`, `age`, `email`, `book_id`, `user_id`, and `book_issue_date`.
     - UNIQUE: On the `email` column in the `library_visitors` table.
     - CHECK: On `count > 0`in `books` and `age >= 18`in `library_visitors`.
3. **Data Manipulation Operations (DML)**
   - Execute the `INSERT INTO`statements to add sample records into all three tables (`books`, `library_visitors`, `book_issue`).
   - Run `SELECT * FROM [table_name]`queries after each insertion to verify the data.
   - Perform an `UPDATE` operation (e.g., changing the email of a user) and verify the change with a `SELECT` statement.
   - Perform a `DELETE` operation (e.g., deleting a book) and ensure that referential integrity is maintained (although no cascading action is explicitly set, the delete should demonstrate the operation).
4. **Access Control and Security**

    - Execute the `CREATE ROLE` command to set up a new user/role named `reporting_user` with a password.
    - Use `GRANT SELECT` statements to give the `reporting_user` read-only access to the main tables (`Departments`, `Employees`, `Projects` - based on the 'Given' section, assuming these are the required reporting tables).
    - Use `REVOKE CREATE` to explicitly prevent the user from creating new database objects within the public schema.
    - Use `REVOKE INSERT, UPDATE, DELETE` to ensure the `reporting_user` has strictly read-only access on all existing tables in the public schema.

5. **Database Maintenance**
    - Execute the `DROP TABLE` command to remove a table that is no longer needed (e.g., `Projects`).

Given:

An organization wants to design a **sample database system** to manage **Departments, Employees, and Projects**. The database must ensure **data integrity**, **controlled access**, and **proper privilege management** for different users.

1. **Database Design**

Create multiple tables such as **Department**, **Employee**, and **Project**.

Define appropriate **PRIMARY KEY** and **FOREIGN KEY** constraints.

Enforce **NOT NULL**, **UNIQUE**, and **CHECK** constraints where necessary.

Querry:

```
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
)
```

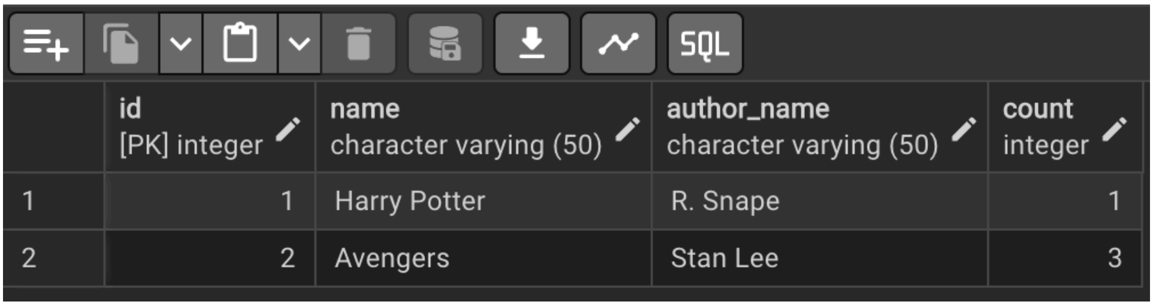
```
CREATE TABLE book_issue(
        book_issue_id INT PRIMARY KEY,
        book_id INT NOT NULL,
        user_id INT NOT NULL,
        FOREIGN KEY (book_id) REFERENCES books(id),
        FOREIGN KEY (user_id) REFERENCES library_visitors(user_id),
        book_issue_date DATE NOT NULL
)
```

2. Data Manipulation

Insert sample records into all tables.

Querry:

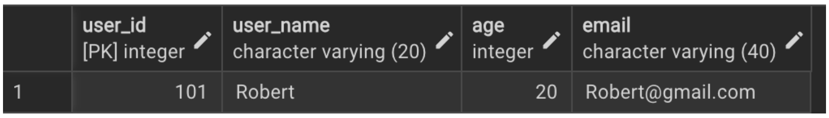
```
INSERT INTO books VALUES(1, 'Harry Potter', 'R. Snape', 1)
INSERT INTO books VALUES(2, 'Avengers', 'Stan Lee', 3)
```



	id [PK] integer	name character varying (50)	author_name character varying (50)	count integer
1	1	Harry Potter	R. Snape	1
2	2	Avengers	Stan Lee	3

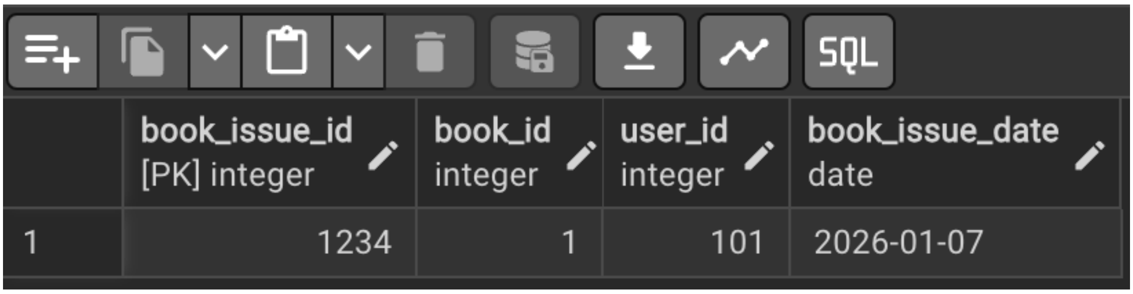
```
SELECT * FROM books
```

```
INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@gmail.com')
```



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

```
SELECT * FROM library_visitors
INSERT INTO book_issue VALUES(1234,1,101,'2026-01-07')
SELECT * FROM book_issue
```



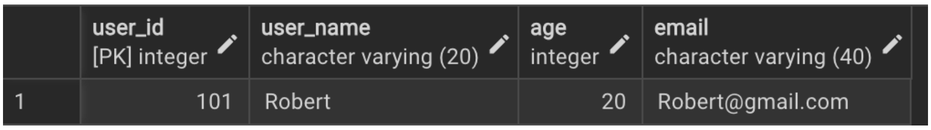
	book_issue_id [PK] integer	book_id integer	user_id integer	book_issue_date date
1	1234	1	101	2026-01-07

Perform **UPDATE** operations to modify existing records.

Querry:

Change an employee's email

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



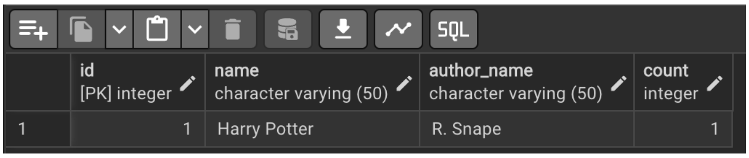
	user_id [PK] integer	user_name character varying (20)	age integer	email character varying (40)
1	101	Robert	20	Robert@gmail.com

Perform **DELETE** operations while maintaining referential integrity.

Querry:

DELETE FROM books WHERE id = 2

SELECT * FROM books



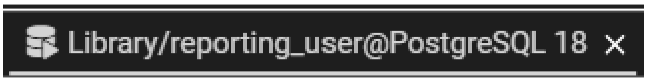
	id [PK] integer	name character varying (50)	author_name character varying (50)	count integer
1	1	Harry Potter	R. Snape	1

3. **Access Control & Security**

Create a **role/user** for a reporting staff member.

Querry:

CREATE ROLE reporting_user LOGIN PASSWORD 'report123';



Library/reporting_user@PostgreSQL 18

Grant **ONLY SELECT privilege** on required tables to this role/user.

Querry:

GRANT SELECT ON Departments TO reporting_user;

GRANT SELECT ON Employees TO reporting_user;

GRANT SELECT ON Projects TO reporting_user;

Explicitly **REVOKE CREATE privilege** so that the user cannot create any database objects.

Querry:

REVOKE CREATE ON SCHEMA public FROM reporting_user;

Ensure the user has **read-only access** to the database.

Querry:

REVOKE INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public FROM reporting_user;

Drop a table that is no longer required using **DROP TABLE**.

Querry:

DROP TABLE Projects;

I/O Analysis

This section analyzes the primary inputs provided to the PostgreSQL database system and the corresponding outputs generated, as executed in the **Procedure** and **Querry** sections.

Component	Input (The Action/Command)	Output (The Expected Result/Status)
Database Design (DDL)	`CREATE TABLE` statements with constraints (PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK).	Successful creation of the `books`, `library_visitors`, and `book_issue` tables. Verification that all defined constraints are active (e.g., no insertion of a book with `count <= 0`).

Data Manipulation (DML)	`INSERT INTO` statements with sample data for all tables.	Successful addition of new records. Data integrity is maintained (e.g., foreign key check ensures only existing user/book IDs can be used in `book_issue`).
	`SELECT * FROM [table_name]` queries.	Display of the newly inserted or modified data sets, confirming the DML operation was successful.
	`UPDATE` operation (e.g., changing a user's email).	One or more rows are modified, and the change is committed to the table. `SELECT` query confirms the new email address.
	`DELETE` operation (e.g., deleting a book).	The specified row is removed. If the book ID was used in `book_issue`, the database prevents the deletion (due to foreign key constraint) unless a cascading action was defined.
Access Control (DCL)	`CREATE ROLE` command.	Successful creation of a new database role named `reporting_user` with a specified password.
	`GRANT SELECT` statements.	The `reporting_user` is successfully assigned read-only access to the specified tables (`Departments`, `Employees`, `Projects`).
	`REVOKE` statements (CREATE, INSERT, UPDATE, DELETE).	Explicit removal of object creation and data modification privileges, ensuring the `reporting_user` has strictly read-only access.
Database Maintenance	`DROP TABLE` command.	Successful removal of the specified table (e.g., `Projects`) from the database schema.

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.