# Project Report

## Project Members and Contributions

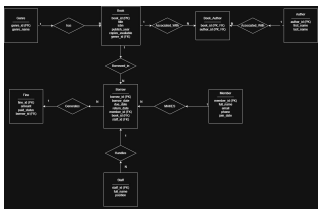
- Tresor: Author and Book data setup
- Sultan: Member data setup
- Zachariah: Report preparation and queries

## 1. Introduction

This Library Database Management System (DBMS) was developed using PostgreSQL for the CMPE343 course. The system manages authors, books, genres, members, staff, borrow transactions, and fines. Assumptions include unique IDs for each entity, proper constraints for data integrity, and realistic sample data.

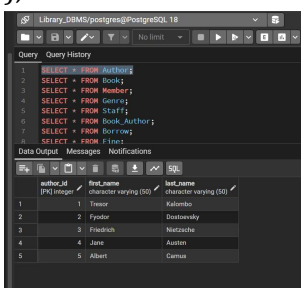
## 2. Database

### ER Diagram

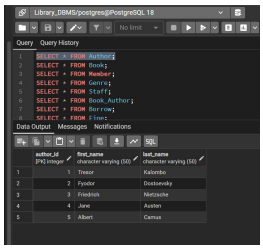


### Data Definition Language (DDL)

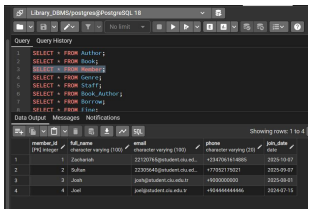
```
CREATE TABLE Author (
   author_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL
);
```



```
CREATE TABLE Book (
   book_id INT PRIMARY KEY,
   title VARCHAR(100) NOT NULL,
   isbn VARCHAR(20) UNIQUE,
   publish_year INT,
   copies_available INT DEFAULT 1 CHECK (copies_available >= 0),
   genre_id INT,
   FOREIGN KEY (genre_id) REFERENCES Genre(genre_id)
);
```

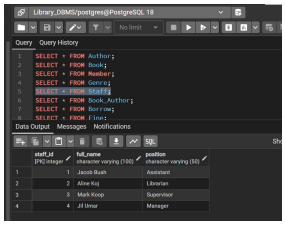


```
CREATE TABLE Member (
   member_id INT PRIMARY KEY,
   full_name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE,
   phone VARCHAR(20),
   join_date DATE DEFAULT CURRENT_DATE
);
```

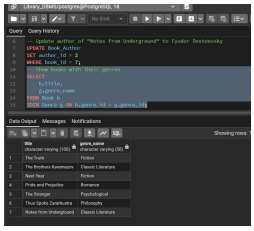


```
CREATE TABLE Staff (
   staff_id INT PRIMARY KEY,
   full_name VARCHAR(100) NOT NULL,
   position VARCHAR(50)
```

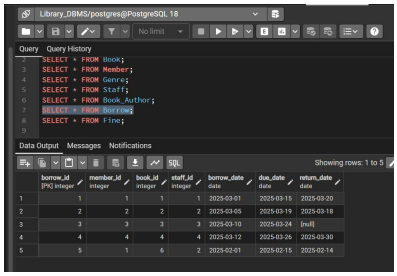
);



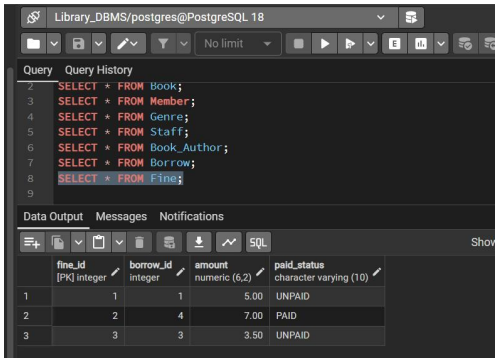
CREATE TABLE Book_Author (
  book_id INT,
  author_id INT,
  PRIMARY KEY (book_id, author_id),
  FOREIGN KEY (book_id) REFERENCES Book(book_id),
  FOREIGN KEY (author_id) REFERENCES Author(author_id)
);



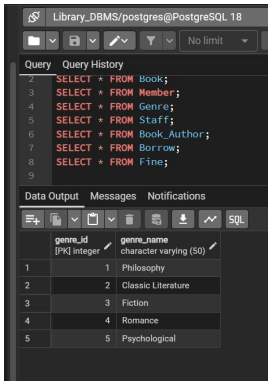
CREATE TABLE Borrow (
  borrow_id INT PRIMARY KEY,
  member_id INT,
  book_id INT,
  staff_id INT,
  borrow_date DATE NOT NULL,
  due_date DATE NOT NULL,
  return_date DATE,
  FOREIGN KEY (member_id) REFERENCES Member(member_id),
  FOREIGN KEY (book_id) REFERENCES Book(book_id),
  FOREIGN KEY (staff_id) REFERENCES Staff(staff_id)
);



```
CREATE TABLE Fine (
   fine_id INT PRIMARY KEY,
   borrow_id INT UNIQUE,
   amount DECIMAL(6,2),
   paid_status VARCHAR(10) DEFAULT 'UNPAID',
   FOREIGN KEY (borrow_id) REFERENCES Borrow(borrow_id)
);
```




```
CREATE TABLE Genre (
   genre_id INT PRIMARY KEY,
   genre_name VARCHAR(50) NOT NULL
);
```



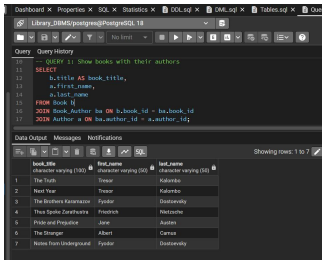

## Data Manipulation Language (DML)

INSERT and UPDATE statements for Author, Book, Book_Author, Member, Staff, Borrow, Fine, Genre are included in the appendix.

## 3. Queries

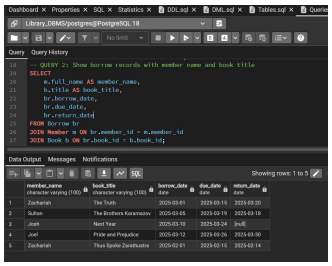
### Query 1

-- QUERY 1: Show books with their authors
SELECT b.title, a.first_name, a.last_name FROM Book b JOIN Book_Author ba ON b.book_id = ba.book_id JOIN Author a ON ba.author_id = a.author_id;



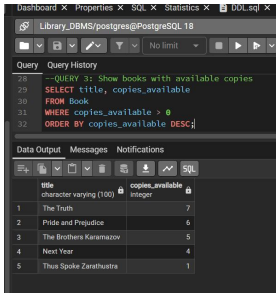
### Query 2

-- QUERY 2: Show borrow records with member name and book title
SELECT m.full_name, b.title, br.borrow_date, br.due_date, br.return_date FROM Borrow br JOIN Member m ON br.member_id = m.member_id JOIN Book b ON br.book_id = b.book_id;



### Query 3

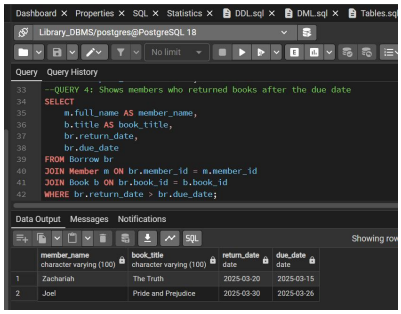
-- QUERY 3: Show books with available copies
SELECT title, copies_available FROM Book WHERE copies_available > 0 ORDER BY copies_available DESC;



### Query 4

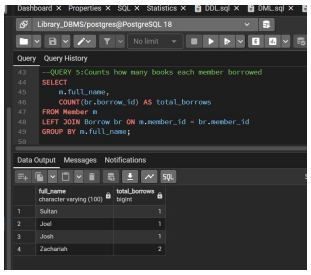
-- QUERY 4: Show members who returned books after the due date
SELECT m.full_name, b.title, br.return_date, br.due_date FROM Borrow br JOIN Member m

ON br.member_id = m.member_id JOIN Book b ON br.book_id = b.book_id WHERE br.return_date > br.due_date;



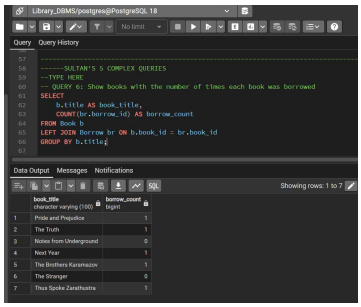

## Query 5

-- QUERY 5: Count how many books each member borrowed
SELECT m.full_name, COUNT(br.borrow_id) AS total_borrows FROM Member m LEFT JOIN Borrow br ON m.member_id = br.member_id GROUP BY m.full_name;



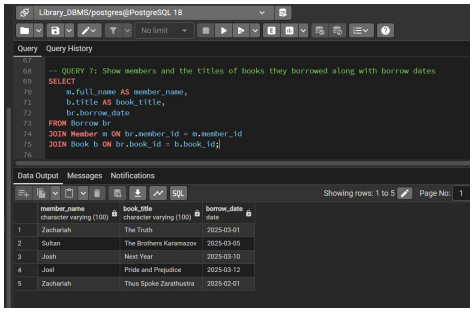

## Query 6

-- QUERY 6: Show books with the number of times each book was borrowed
SELECT b.title AS book_title, COUNT(br.borrow_id) AS borrow_count FROM Book b LEFT JOIN Borrow br ON b.book_id = br.book_id GROUP BY b.title;



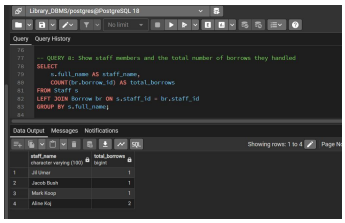
## Query 7

-- QUERY 7: Show members and the titles of books they borrowed along with borrow dates
SELECT m.full_name AS member_name, b.title AS book_title, br.borrow_date FROM Borrow br JOIN Member m ON br.member_id = m.member_id JOIN Book b ON br.book_id = b.book_id;



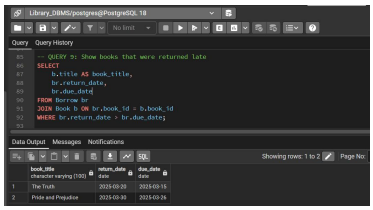

## Query 8

-- QUERY 8: Show staff members and the total number of borrows they handled
SELECT s.full_name AS staff_name, COUNT(br.borrow_id) AS total_borrows FROM Staff s LEFT JOIN Borrow br ON s.staff_id = br.staff_id GROUP BY s.full_name;



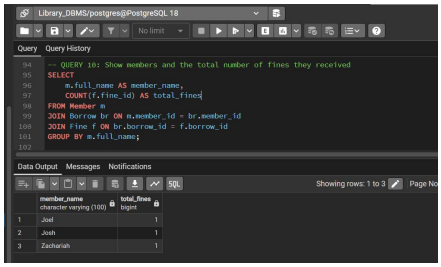

## Query 9

-- QUERY 9: Show books that were returned late
SELECT b.title AS book_title, br.return_date, br.due_date FROM Borrow br JOIN Book b ON br.book_id = b.book_id WHERE br.return_date > br.due_date;



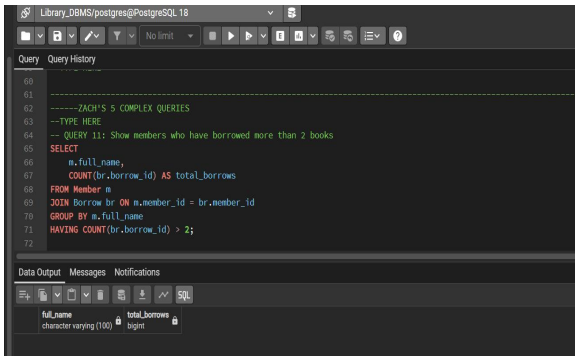
## Query 10

-- QUERY 10: Show members and the total number of fines they received
SELECT m.full_name AS member_name, COUNT(f.fine_id) AS total_fines FROM Member m JOIN Borrow br ON m.member_id = br.member_id JOIN Fine f ON br.borrow_id = f.borrow_id GROUP BY m.full_name;



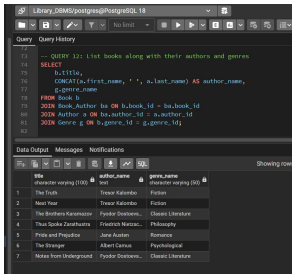

## Query 11

-- QUERY 11: Show members who borrowed more than 2 books
SELECT m.full_name, COUNT(br.borrow_id) AS total_borrows FROM Member m JOIN Borrow br ON m.member_id = br.member_id GROUP BY m.full_name HAVING COUNT(br.borrow_id) > 2;



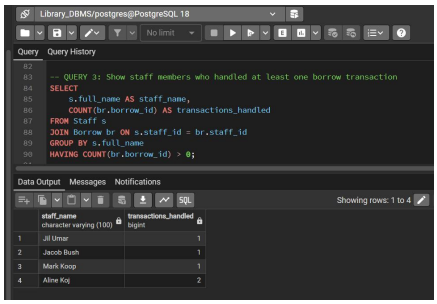

## Query 12

-- QUERY 12: List books along with their authors and genres
SELECT b.title, CONCAT(a.first_name, ' ', a.last_name) AS author_name, g.genre_name FROM Book b JOIN Book_Author ba ON b.book_id = ba.book_id JOIN Author a ON ba.author_id = a.author_id JOIN Genre g ON b.genre_id = g.genre_id;



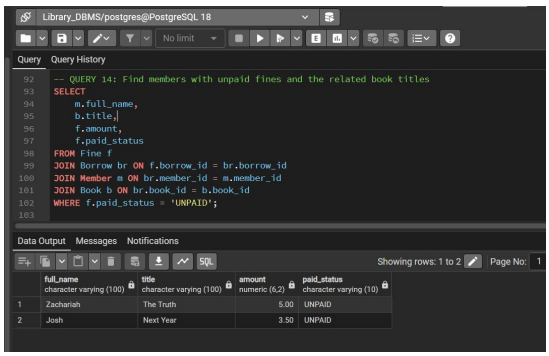
## Query 13

-- QUERY 13: Show staff members who handled at least one borrow transaction
SELECT s.full_name AS staff_name, COUNT(br.borrow_id) AS transactions_handled FROM Staff s JOIN Borrow br ON s.staff_id = br.staff_id GROUP BY s.full_name HAVING COUNT(br.borrow_id) > 0;




## Query 14

-- QUERY 14: Find members with unpaid fines and the related book titles
SELECT m.full_name, b.title, f.amount, f.paid_status FROM Fine f JOIN Borrow br ON f.borrow_id = br.borrow_id JOIN Member m ON br.member_id = m.member_id JOIN Book b ON br.book_id = b.book_id WHERE f.paid_status = 'UNPAID';




## Query 15

-- QUERY 15: Show the most popular genre based on number of borrows
SELECT g.genre_name, COUNT(br.borrow_id) AS borrow_count FROM Genre g JOIN Book b ON g.genre_id = b.genre_id JOIN Borrow br ON b.book_id = br.book_id GROUP BY g.genre_name ORDER BY borrow_count DESC LIMIT 1;