

Project Report

Project Members and Contributions

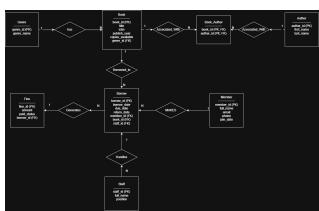
- Tresor: Author and Book data setup
- Sultan: Member data setup
- Zachariah: Report preparation and queries

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
);
```

The screenshot shows the PostgreSQL client interface with the following content:

- Query:** A list of SQL commands:

```
1. SELECT * FROM Authors;
2. SELECT * FROM Books;
3. SELECT * FROM Member;
4. SELECT * FROM Genre;
5. SELECT * FROM Staff;
6. SELECT * FROM Borrow;
7. SELECT * FROM Fines;
```
- Data Output:** A table showing the data for the Authors table:

author_id	first_name	last_name
1	Fyodor	Dostoevsky
2	Albert	Camus
3	Jane	Austen
4	Friedrich	Nietzsche

```

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)
);

```

author_id	first_name	last_name
1	Tresor	Kalimbo
2	Michael	Obi
3	Frederick	Nwachukwu
4	Jane	Austen
5	Albert	Camus

```

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
);

```

book_id	title	isbn	phone	join_date
1	Zikrahah	2235076@student.us.edu	+234706114805	2025-10-07
2	Obafemi	2235054@student.us.edu	+234706114807	2025-10-08
3	John	4563000000	+919999999999	2025-10-08
4	Joel	jun@student.us.edu	+919999999999	2024-07-15

```

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

```

);

The screenshot shows the pgAdmin 4 interface with a query editor window. The code in the editor is:

```
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)
);
```

Below the editor, the Data Output tab shows the results of a SELECT query on the Book_Author table:

staff_id	full_name	character_varying (100)	character_varying (10)
1	Jacob Bush	Assistant	
2	Alice Kye	Librarian	
3	Bob Hope	Internship	
4	JJ Turner	Manager	

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)
);
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The code in the editor is:

```
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)
);
```

Below the editor, the Data Output tab shows the results of a SELECT query on the Borrow table:

borrow_id	member_id	book_id	staff_id	borrow_date	due_date	return_date
1	1	1	1	2025-03-01	2025-03-15	2025-03-20
2	2	2	2	2025-03-05	2025-03-19	2025-03-18
3	3	3	3	2025-03-10	2025-03-24	[null]
4	4	4	4	2025-03-12	2025-03-26	2025-03-30
5	5	1	6	2025-02-01	2025-02-15	2025-02-14

CREATE TABLE Book (

```
    title character varying (100),
    genre character varying (50),
    publisher character varying (50),
    year integer,
    price numeric(10,2),
    FOREIGN KEY (genre) REFERENCES Genre(genre),
    FOREIGN KEY (publisher) REFERENCES Publisher(publisher),
    FOREIGN KEY (year) REFERENCES Year(year),
    FOREIGN KEY (price) REFERENCES Price(price)
);
```

The screenshot shows the pgAdmin 4 interface with a query editor window. The code in the editor is:

```
CREATE TABLE Book (
    title character varying (100),
    genre character varying (50),
    publisher character varying (50),
    year integer,
    price numeric(10,2),
    FOREIGN KEY (genre) REFERENCES Genre(genre),
    FOREIGN KEY (publisher) REFERENCES Publisher(publisher),
    FOREIGN KEY (year) REFERENCES Year(year),
    FOREIGN KEY (price) REFERENCES Price(price)
);
```

Below the editor, the Data Output tab shows the results of a SELECT query on the Book table:

title	genre	publisher	year	price
The Craft	Fiction	Author Publishing	2025	12.99
The Great Resuscitation	Classical literature	Author Publishing	2025	14.99
New Hope	Romance	Author Publishing	2025	15.99
Pride and Prejudice	Romance	Author Publishing	2025	16.99
The Last Stand	Thriller	Author Publishing	2025	17.99
True Spy: Zenithians	Philosophy	Author Publishing	2025	18.99
Note from Underground	Classical literature	Author Publishing	2025	19.99

```

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)
);

```

The screenshot shows the pgAdmin interface with a query editor containing the SQL code for creating the Fine table. Below the editor is a data grid displaying three rows of data:

	fine_id	borrow_id	amount	paid_status
1	1	1	5.00	UNPAID
2	2	4	7.00	PAID
3	3	3	3.50	UNPAID

```

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

```

The screenshot shows the pgAdmin interface with a query editor containing the SQL code for creating the Genre table. Below the editor is a data grid displaying five rows of data:

	genre_id	genre_name
1	1	Philosophy
2	2	Classic Literature
3	3	Fiction
4	4	Romance
5	5	Psychological

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;
```

book_title	first_name	last_name
The Truth	Tessie	Kalvano
New Year	Tessie	Kalvano
The Brothers Karamazov	Stephanie	Stepanov
Thus Spoke Zarathustra	Friedrich	Nietzsche
Pride and Prejudice	Jane	Austen
The Stranger	Albert	Camus
Nostradamus	Sybil	Babynousky

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;
```

full_name	title	borrow_date	due_date	return_date
Zachariah	The Truth	2025-03-10	2025-03-15	2025-03-30
Joel	The Brothers Karamazov	2025-03-10	2025-03-26	2025-04-10
John	New Year	2025-03-10	2025-03-26	2025-04-06
Joel	Pride and Prejudice	2025-03-12	2025-03-26	2025-04-06
Zachariah	Thus Spoke Zarathustra	2025-02-01	2025-02-15	2025-03-14

Query 3

-- QUERY 3: Show books with available copies

```
SELECT title, copies_available FROM Book WHERE copies_available > 0 ORDER BY copies_available DESC;
```

title	copies_available
The Truth	7
Pride and Prejudice	6
The Brothers Karamazov	5
New Year	4
Thus Spoke Zarathustra	1

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 4: Shows members who returned books after the due date
SELECT
  m.full_name AS member_name,
  b.title AS book_title,
  br.return_date,
  br.due_date
FROM Member m
JOIN Book b ON br.member_id = m.member_id
JOIN Borrow br ON br.book_id = b.book_id
WHERE br.return_date > br.due_date;
```

member_name	book_title	return_date	due_date
Zachariah	The Truth	2025-03-20	2025-03-15
Joel	Pride and Prejudice	2025-03-30	2025-03-26

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 5:Counts 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;
```

full_name	total_borrows
Sultan	1
Joel	1
John	1
Zachariah	2

Query 6

-- QUERY 6: Show books with the number of times each book was borrowed

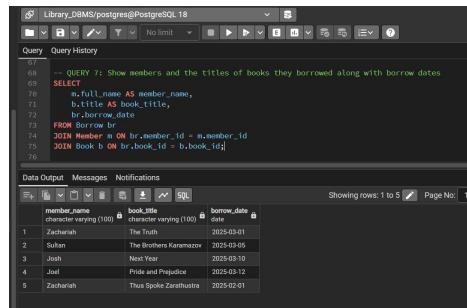
```
SELECT b.title AS book_title, COUNT(br.borrow_id) AS borrow_count FROM Book b LEFT
JOIN Borrow br ON b.book_id = br.book_id GROUP BY b.title;
```

```
-- SULTAN'S 5 COMPLEX QUERIES
-- TYPE HERE
-- 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;
```

book_title	borrow_count
Pride and Prejudice	1
The Truth	1
Notes from Underground	0
Next Year	1
The Brothers Karamazov	1
The Stranger	0
Thus Spake Zarathustra	1

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;



The screenshot shows a PostgreSQL 18 query editor window. The query is:

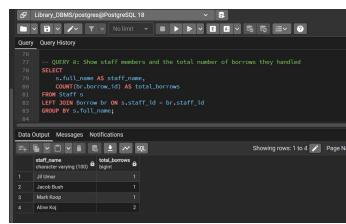
```
-- QUERY 7: Show members and the titles of books they borrowed along with borrow dates
SELECT
    m.full_name AS member_name,
    b.title AS book_title,
    br.borrow_date
FROM Borrow br
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id;
```

The results table has columns: member_name, book_title, and borrow_date. The data is:

member_name	book_title	borrow_date
Zachariah	The Truth	2025-03-01
Sultan	The Brothers Karamazov	2025-03-05
Josh	New Year	2025-03-10
Joel	Pride and Prejudice	2025-03-12
Zachariah	Thus Spoke Zarathustra	2025-02-01

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;



The screenshot shows a PostgreSQL 18 query editor window. The query is:

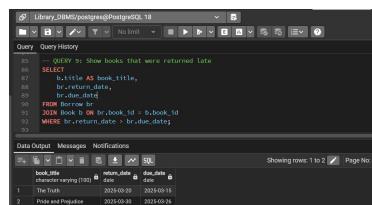
```
-- QUERY 8: Show staff members and the total number of borrows they handled
SELECT
    s.full_name AS staff_name,
    COUNT(br.borrow_id) AS total_borrows
FROM Staff s
LEFT JOIN Borrow br ON s.staff_id = br.staff_id
GROUP BY s.full_name;
```

The results table has columns: staff_name and total_borrows. The data is:

staff_name	total_borrows
Jill Turner	1
Joel Bush	1
Alice May	2

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;



The screenshot shows a PostgreSQL 18 query editor window. The query is:

```
-- 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;
```

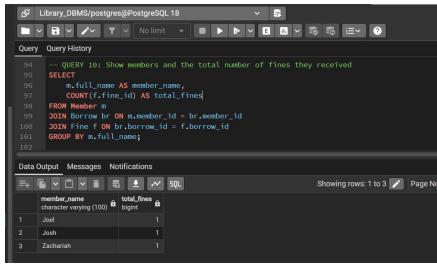
The results table has columns: book_title, return_date, and due_date. The data is:

book_title	return_date	due_date
The Truth	2025-03-30	2025-03-15
Pride and Prejudice	2025-03-30	2025-03-26

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;
```



The screenshot shows a PostgreSQL query editor window. The query is:

```
-- 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;
```

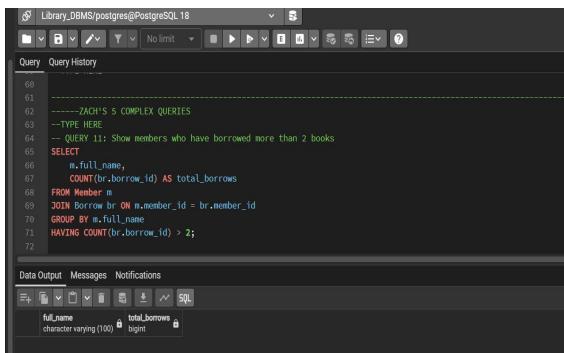
The results table has columns: member_name (character varying(100)) and total_fines (integer). The data is:

member_name	total_fines
Josh	1
Zachariah	1

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;
```



The screenshot shows a PostgreSQL query editor window. The query is:

```
-- ZACH'S 5 COMPLEX QUERIES
-- TYPE HERE
-- QUERY 11: Show members who have 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;
```

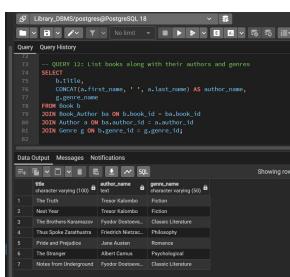
The results table has columns: fullName (character varying(100)) and totalborrows (integer). The data is:

fullName	totalborrows
Josh	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;
```



The screenshot shows a PostgreSQL query editor window. The query is:

```
-- 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;
```

The results table has columns: title (character varying(100)), author_name (character varying(60)), and genre_name (character varying(60)). The data is:

title	author_name	genre_name
The Brothers Karamazov	Fyodor Dostoevsky	Classics
Non-Stop	Trevor Kalinick	Fiction
The Brothers Karamazov	Fyodor Dostoevsky	Classics
The Spidey Zenturions	Freddie Newmeyer	Philosophy
The Brothers Karamazov	Fyodor Dostoevsky	Classics
The Stranger	Albert Camus	Psychological
Noise from Underground	Fyodor Dostoevsky	Classics

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;
```

staff_name	transactions_handled
Jill Umar	1
Jacobo Bush	1
Mark Koop	1
Aline Koj	2

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';
```

full_name	title	amount	paid_status
Zachariah	The Truth	5.00	UNPAID
Josh	Next Year	3.50	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;
```

genre_name	borrow_count
Fiction	2