

Project Report

Project Members and Contributions

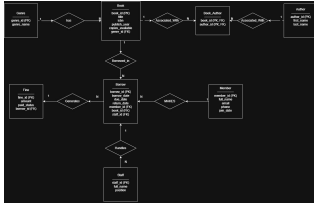
- Tresor: Author and Book data setup
- Sultan: Member data setup
- Zachariah: Report preparation and queries

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



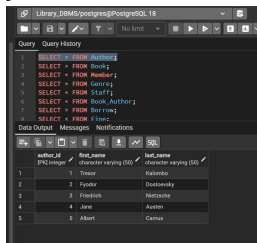
Data Definition Language (DDL)

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

The screenshot shows a PostgreSQL query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays a list of queries, with the first query being the DDL statement for the Author table. The Data Output pane shows the results of a query executed against the Author table, displaying a table with columns 'author_id', 'first_name', and 'last_name'.

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

```
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)
);
```



The screenshot shows a PostgreSQL query history window with the following queries:

```

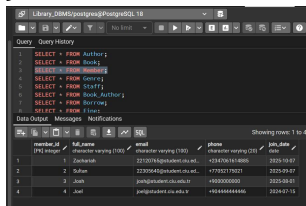
1 SELECT * FROM Book;
2 SELECT * FROM Author;
3 SELECT * FROM Genre;
4 SELECT * FROM Staff;
5 SELECT * FROM Book_Author;
6 SELECT * FROM Reviews;
7 SELECT * FROM User;

```

The data output window shows the following data for the Book table:

book_id	title	isbn	publish_year	copies_available	genre_id
1	Twilight	9780307266666	2005	10	1
2	Twilight	9780307266666	2005	10	1
3	Twilight	9780307266666	2005	10	1
4	Twilight	9780307266666	2005	10	1
5	Twilight	9780307266666	2005	10	1

```
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
);
```



The screenshot shows a PostgreSQL query history window with the following queries:

```

1 SELECT * FROM Author;
2 SELECT * FROM Book;
3 SELECT * FROM Genre;
4 SELECT * FROM Staff;
5 SELECT * FROM Book_Author;
6 SELECT * FROM Reviews;
7 SELECT * FROM User;

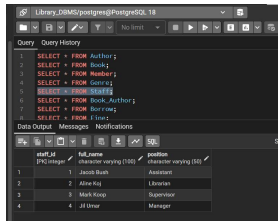
```

The data output window shows the following data for the Member table:

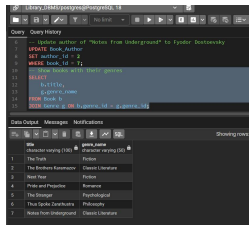
member_id	full_name	email	phone	join_date
1	John Doe	john.doe@example.com	+1234567890	2023-10-27
2	Jane Doe	jane.doe@example.com	+1234567890	2023-10-27
3	John Doe	john.doe@example.com	+1234567890	2023-10-27
4	John Doe	john.doe@example.com	+1234567890	2023-10-27

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

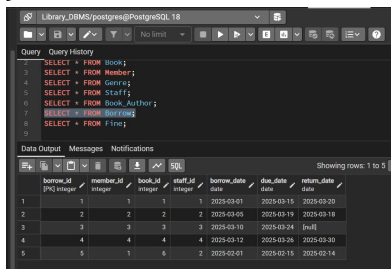
);



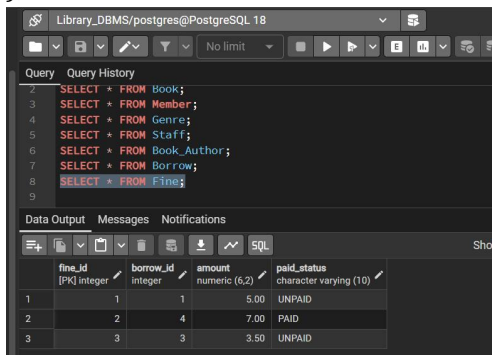
```
CREATE TABLE Book_Author (
    book_id INT,
    author_id INT,
    PRIMARY KEY (book_id, author_id),
    FOREIGN KEY (book_id) REFERENCES Book(book_id),
    FOREIGN KEY (author_id) REFERENCES Author(author_id)
);
```



```
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)
);
```



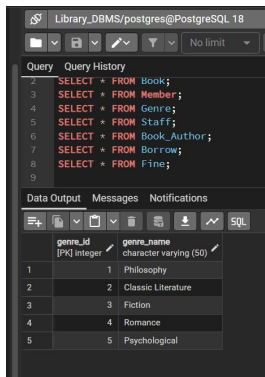
```
CREATE TABLE Fine (
    fine_id INT PRIMARY KEY,
    borrow_id INT UNIQUE,
    amount DECIMAL(6,2),
    paid_status VARCHAR(10) DEFAULT 'UNPAID',
    FOREIGN KEY (borrow_id) REFERENCES Borrow(borrow_id)
);
```



The screenshot shows a PostgreSQL query editor with a query history list and a data output table. The query history list contains several 'SELECT * FROM' statements for tables: Book, Member, Genre, Staff, Book_Author, Borrow, and Fine. The data output table displays the contents of the 'Fine' table, which has four columns: fine_id (PK integer), borrow_id (integer), amount (numeric (6,2)), and paid_status (character varying (10)).

fine_id [PK] integer	borrow_id integer	amount numeric (6,2)	paid_status character varying (10)
1	1	5.00	UNPAID
2	2	7.00	PAID
3	3	3.50	UNPAID

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



The screenshot shows a PostgreSQL query editor with a query history list and a data output table. The query history list contains several 'SELECT * FROM' statements for tables: Book, Member, Genre, Staff, Book_Author, Borrow, and Fine. The data output table displays the contents of the 'Genre' table, which has two columns: genre_id (PK integer) and genre_name (character varying (50)).

genre_id [PK] integer	genre_name character varying (50)
1	Philosophy
2	Classic Literature
3	Fiction
4	Romance
5	Psychological

Data Manipulation Language (DML)

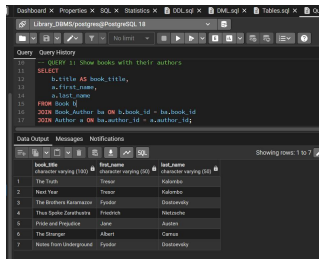
INSERT and UPDATE statements for Author, Book, Book_Author, Member, Staff, Borrow, Fine, Genre are included in the appendix.

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;
```



The screenshot shows a database query editor with the following SQL query:

```
-- QUERY 1: Show books with their authors  
SELECT  
1 b.title AS book_title,  
2 a.first_name,  
3 a.last_name  
4 FROM Book b  
5 JOIN Book_Author ba ON b.book_id = ba.book_id  
6 JOIN Author a ON ba.author_id = a.author_id;
```

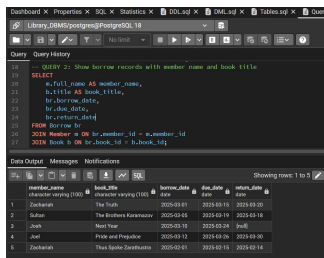
The results are displayed in a table with 3 columns: book_id, title, and author. The data is as follows:

book_id	title	author
1	The Hobbit	J.R.R. Tolkien
2	The Lord of the Rings	J.R.R. Tolkien
3	The Hobbit	J.R.R. Tolkien
4	The Hobbit	J.R.R. Tolkien
5	The Hobbit	J.R.R. Tolkien
6	The Hobbit	J.R.R. Tolkien
7	The Hobbit	J.R.R. Tolkien

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;
```



The screenshot shows a database query editor with the following SQL query:

```
-- QUERY 2: Show borrow records with member name and book title  
SELECT  
1 m.full_name AS member_name,  
2 b.title AS book_title,  
3 br.borrow_date,  
4 br.due_date,  
5 br.return_date  
6 FROM Borrow br  
7 JOIN Member m ON br.member_id = m.member_id  
8 JOIN Book b ON br.book_id = b.book_id;
```

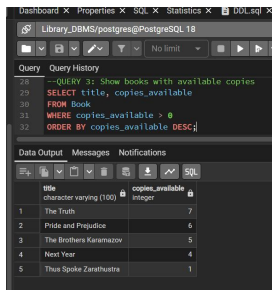
The results are displayed in a table with 5 columns: member_name, book_title, borrow_date, due_date, and return_date. The data is as follows:

member_name	book_title	borrow_date	due_date	return_date
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01
J.R.R. Tolkien	The Hobbit	2020-03-01	2020-03-15	2020-03-01

Query 3

-- QUERY 3: Show books with available copies

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



The screenshot shows a database query editor with the following SQL query:

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

The results are displayed in a table with 2 columns: title and copies_available. The data is as follows:

title	copies_available
The Hobbit	7
The Lord of the Rings	6
The Hobbit	5
The Hobbit	4
The Hobbit	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 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;

```

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
Sutton	1
Joel	1
Joah	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;

Query 6: Show books with the number of times each book was borrowed.

```

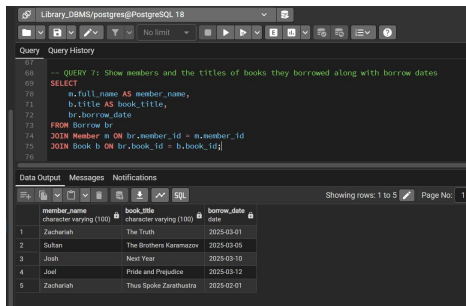
SELECT
  b.title AS book_title,
  COUNT(br.borrow_id) AS borrow_count
FROM Book b
LEFT JOIN Borrow br ON b.book_id = br.book_id
GROUP BY b.title;

```

book_title	borrow_count
Pride and Prejudice	1
The Truth	1
Mosses from Underground	0
Next Year	1
The Bottom Reason	1
The Stranger	0
Thou Spoke Zachariah	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 query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the following SQL query:

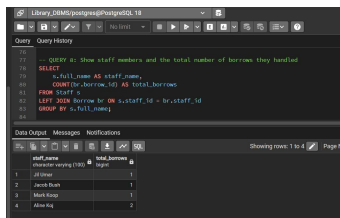
```
-- QUERY 7: Show members and the titles of books they borrowed along with borrow dates
SELECT
  m.full_name AS member_name,
  b.title AS book_title,
  br.borrow_date
FROM Borrow br
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id;
```

The Data Output pane shows the results of the query, displaying 5 rows. The columns are member_name, book_title, and borrow_date.

member_name	book_title	borrow_date
Zachariah	The Truth	2025-02-01
Salma	The Brothers Karamazov	2025-03-08
Joah	Next Year	2025-03-10
Juul	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 query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the following SQL query:

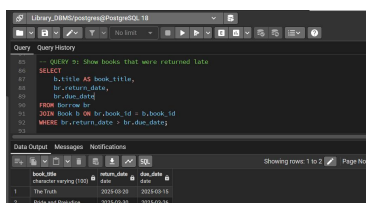
```
-- QUERY 8: Show staff members and the total number of borrows they handled
SELECT
  s.full_name AS staff_name,
  COUNT(br.borrow_id) AS total_borrows
FROM Staff s
LEFT JOIN Borrow br ON s.staff_id = br.staff_id
GROUP BY s.full_name;
```

The Data Output pane shows the results of the query, displaying 4 rows. The columns are staff_name and total_borrows.

staff_name	total_borrows
Joah	1
Salma	1
Joah	1
Salma	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 query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the following SQL query:

```
-- 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;
```

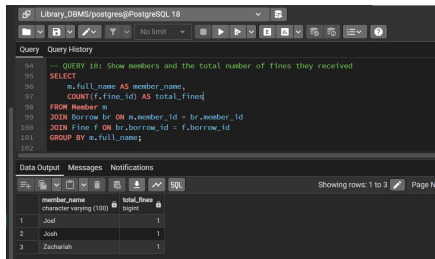
The Data Output pane shows the results of the query, displaying 2 rows. The columns are book_title, return_date, and due_date.

book_title	return_date	due_date
The Truth	2025-03-01	2025-03-10
Pride and Prejudice	2025-03-01	2025-03-10

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;
```



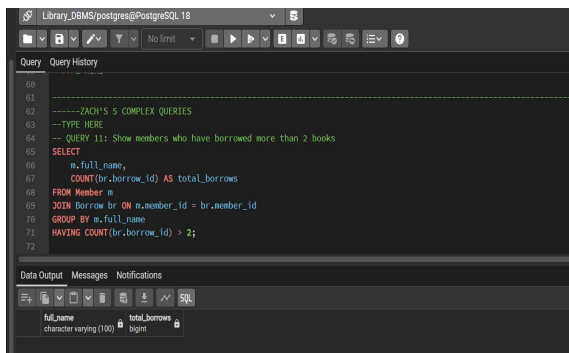
The screenshot shows a PostgreSQL query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the SQL for Query 10. The data output pane shows the results of the query, which are three rows of member names and their total fines.

member_name	total_fines
Jodi	1
Joah	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;
```



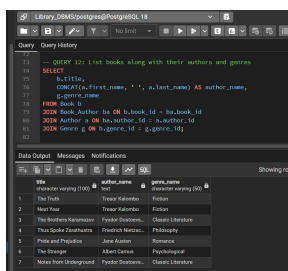
The screenshot shows a PostgreSQL query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the SQL for Query 11. The data output pane shows the results of the query, which are three rows of member names and their total borrows.

full_name	total_borrows
Zachariah	3
Joah	2
Jodi	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;
```

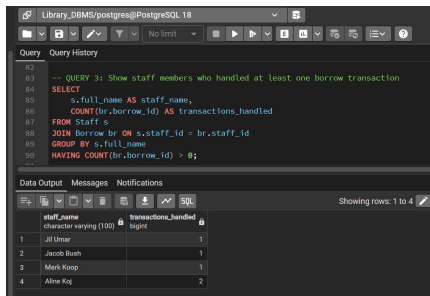


The screenshot shows a PostgreSQL query editor window titled 'Library_DBMS/postgres@PostgreSQL 18'. The query history pane displays the SQL for Query 12. The data output pane shows the results of the query, which are seven rows of book titles, author names, and genre names.

title	author_name	genre_name
The Fault	Travis Alabaster	Fiction
New York	Travis Alabaster	Fiction
The Redneck's Companion	Franklin D. Roosevelt	Classics Literature
The Spide Zandwilde	Franklin D. Roosevelt	Philosophy
Philosophy and Psychology	John Doe	Science
The Stranger	Albert Camus	Psychological
Notes from Underground	Fyodor Dostoevsky	Classics Literature

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;



The screenshot shows a PostgreSQL query editor with the following SQL query:

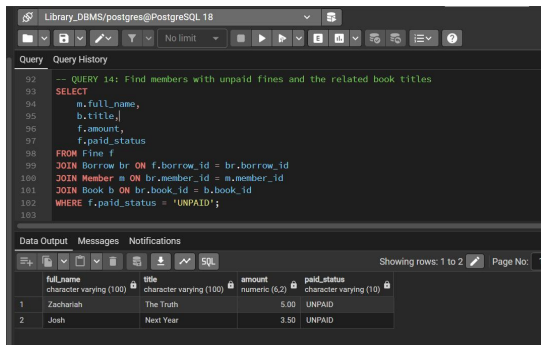
```
-- QUERY 13: Show staff members who handled at least one borrow transaction
SELECT
  s.full_name AS staff_name,
  COUNT(br.borrow_id) AS transactions_handled
FROM Staff s
JOIN Borrow br ON s.staff_id = br.staff_id
GROUP BY s.full_name
HAVING COUNT(br.borrow_id) > 0;
```

The Data Output tab shows the following results:

staff_name	transactions_handled
John Doe	1
Jacob Smith	1
Mark King	1
Alice King	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';



The screenshot shows a PostgreSQL query editor with the following SQL query:

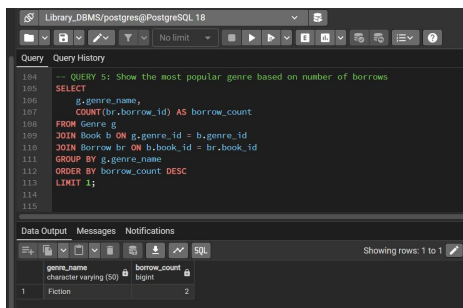
```
-- QUERY 14: Find members with unpaid fines and the related book titles
SELECT
  m.full_name,
  b.title,
  f.amount,
  f.paid_status
FROM Fine f
JOIN Borrow br ON f.borrow_id = br.borrow_id
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id
WHERE f.paid_status = 'UNPAID';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

The Data Output tab shows the following results:

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;



The screenshot shows a PostgreSQL query editor with the following SQL query:

```
-- 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;
```

The Data Output tab shows the following results:

genre_name	borrow_count
Fiction	2