Project Report

Project Members and Contributions

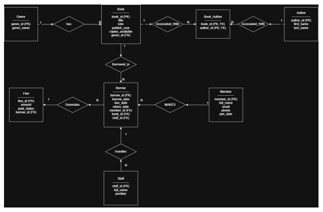
- Tresor: Author and Book data setup
- Sultan: Member data setup
- Zachariah: Report preparation and queries

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.

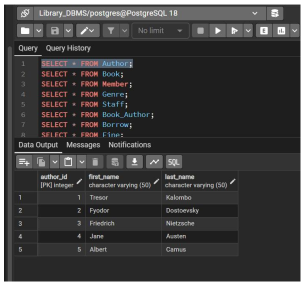
2. Database

ER Diagram



Data Definition Language (DDL)

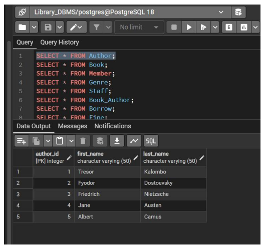
```
CREATE TABLE Author (
   author_id INT PRIMARY KEY,
   first_name VARCHAR(50) NOT NULL,
   last_name VARCHAR(50) NOT NULL
);
```



```
SELECT * FROM Author;
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

	author_id [PK] integer	first_name character varying (50)	last_name character varying (50)
1	1	Tresor	Kalombo
2	2	Fyodor	Dostoevsky
3	3	Friedrich	Nietzsche
4	4	Jane	Austen
5	5	Albert	Camus

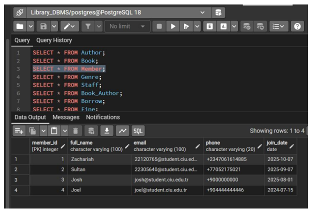
```
CREATE TABLE Book (
   book_id INT PRIMARY KEY,
   title VARCHAR(100) NOT NULL,
   isbn VARCHAR(20) UNIQUE,
   publish_year INT,
   copies_available INT DEFAULT 1 CHECK (copies_available >= 0),
   genre_id INT,
   FOREIGN KEY (genre_id) REFERENCES Genre(genre_id)
);
```



```
SELECT * FROM Author;
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

	author_id [PK] integer	first_name character varying (50)	last_name character varying (50)
1	1	Tresor	Kalombo
2	2	Fyodor	Dostoevsky
3	3	Friedrich	Nietzsche
4	4	Jane	Austen
5	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
);
```

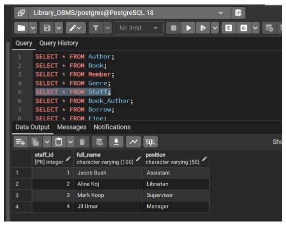


```
SELECT * FROM Author;
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

Showing rows: 1 to 4

	member_id [PK] integer	full_name character varying (100)	email character varying (100)	phone character varying (20)	join_date date
1	1	Zachariah	22120765@student.ciu.ed...	+2347061614885	2025-10-07
2	2	Sultan	22305640@student.ciu.ed...	+77052175021	2025-09-07
3	3	Josh	josh@student.ciu.edu.tr	+9000000000	2025-08-01
4	4	Joel	joel@student.ciu.edu.tr	+904444444446	2024-07-15

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



Library_DBMS/postgres@PostgreSQL 18

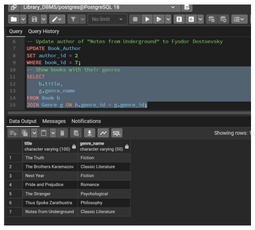
Query | Query History

```
SELECT * FROM Author;
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

Data Output | Messages | Notifications

	staff_id [PK] integer	full_name character varying (100)	position character varying (50)
1	1	Jacob Bush	Assistant
2	2	Aline Koj	Librarian
3	3	Mark Koop	Supervisor
4	4	Jil Umar	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)
);
```



Library_DBMS/postgres@PostgreSQL 18

Query | Query History

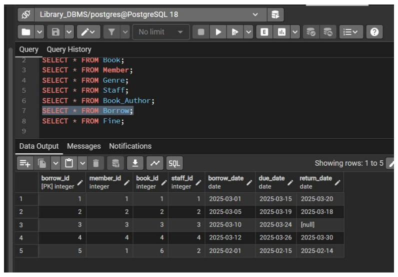
```
-- Update author of "Notes from Underground" to Fyodor Dostoevsky
UPDATE Book_Author
SET author_id = 2
WHERE book_id = 7;
-- Show books with their genres
SELECT
    b.title,
    g.genre_name
FROM Book b
JOIN Genre g ON b.genre_id = g.genre_id;
```

Data Output | Messages | Notifications

Showing rows: 1

	title character varying (100)	genre_name character varying (50)
1	The Truth	Fiction
2	The Brothers Karamazov	Classic Literature
3	Next Year	Fiction
4	Pride and Prejudice	Romance
5	The Stranger	Psychological
6	Thus Spoke Zarathustra	Philosophy
7	Notes from Underground	Classic Literature

```
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)
);
```



Library_DBMS/postgres@PostgreSQL 18

Query | Query History

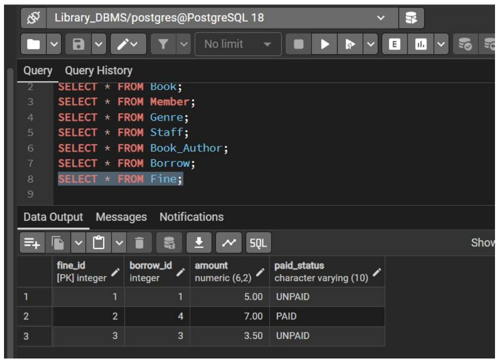
```
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

Data Output | Messages | Notifications

Showing rows: 1 to 5

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

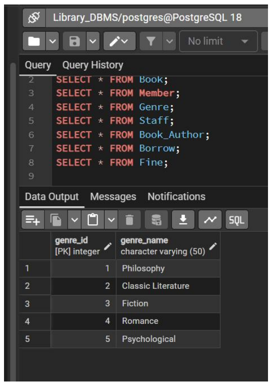
```
CREATE TABLE Fine (
   fine_id INT PRIMARY KEY,
   borrow_id INT UNIQUE,
   amount DECIMAL(6,2),
   paid_status VARCHAR(10) DEFAULT 'UNPAID',
   FOREIGN KEY (borrow_id) REFERENCES Borrow(borrow_id)
);
```



```
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

	fine_id [PK] integer	borrow_id integer	amount numeric (6,2)	paid_status character varying (10)
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
);
```



```
SELECT * FROM Book;
SELECT * FROM Member;
SELECT * FROM Genre;
SELECT * FROM Staff;
SELECT * FROM Book_Author;
SELECT * FROM Borrow;
SELECT * FROM Fine;
```

	genre_id [PK] integer	genre_name character varying (50)
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.

```
INSERT INTO Author VALUES (1, 'Tresor', 'Kalombo');

INSERT INTO Author VALUES (2, 'Fyodor', 'Dostoevsky');

INSERT INTO Author VALUES (3, 'Friedrich', 'Nietzsche');

INSERT INTO Author VALUES (4, 'Jane', 'Austen');
INSERT INTO Author VALUES (5, 'Albert', 'Camus');


-- Books

INSERT INTO Book VALUES (1, 'The Truth', '000077770000', 2057, 7);

INSERT INTO Book VALUES (2, 'The Brothers Karamazov', '000077770001', 1880, 5);

INSERT INTO Book VALUES (3, 'Next Year', '000077770002', 2059, 4);

INSERT INTO Book VALUES (4, 'Pride and Prejudice', '000077770003', 1813, 6);

INSERT INTO Book VALUES (5, 'The Stranger', '000077770004', 1942, 0); -- no copies
available

INSERT INTO Book VALUES (6, 'Thus Spoke Zarathustra', '000077770005', 1883, 1);

INSERT INTO Book VALUES (7, 'Notes from Underground', '000077770006', 1864, 0); -- no
copies available


-- Connect books with authors

-- Author 1 wrote two books

INSERT INTO Book_Author VALUES (1, 1);

INSERT INTO Book_Author VALUES (3, 1);

-- Author 2 wrote two books

INSERT INTO Book_Author VALUES (2, 2);

INSERT INTO Book_Author VALUES (7, 2);

-- Other authors

INSERT INTO Book_Author VALUES (6, 3); -- Nietzsche

INSERT INTO Book_Author VALUES (4, 4); -- Austen

INSERT INTO Book_Author VALUES (5, 5); -- Camus


---Genre

-- Add genre reference to Book table
```

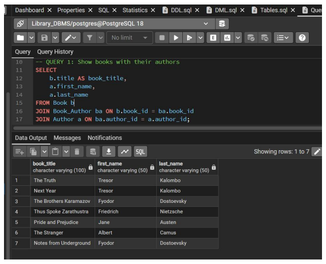
```
ALTER TABLE Book

ADD COLUMN genre_id INT;

-- Add foreign key constraint

ALTER TABLE Book

ADD CONSTRAINT fk_book_genre

FOREIGN KEY (genre_id) REFERENCES Genre(genre_id);

-- Insert genres

INSERT INTO Genre VALUES (1, 'Philosophy');

INSERT INTO Genre VALUES (2, 'Classic Literature');

INSERT INTO Genre VALUES (3, 'Fiction');

INSERT INTO Genre VALUES (4, 'Romance');

INSERT INTO Genre VALUES (5, 'Psychological');

-- Assign genres to books

UPDATE Book SET genre_id = 3 WHERE book_id = 1; -- The Truth (Fiction)

UPDATE Book SET genre_id = 2 WHERE book_id = 2; -- The Brothers Karamazov (Classic)

UPDATE Book SET genre_id = 3 WHERE book_id = 3; -- Next Year (Fiction)

UPDATE Book SET genre_id = 4 WHERE book_id = 4; -- Pride and Prejudice (Romance)

UPDATE Book SET genre_id = 5 WHERE book_id = 5; -- The Stranger (Psychological)

UPDATE Book SET genre_id = 1 WHERE book_id = 6; -- Thus Spoke Zarathustra (Philosophy)

UPDATE Book SET genre_id = 2 WHERE book_id = 7; -- Notes from Underground (Classic)




-- Members

INSERT INTO Member VALUES (1, 'Zachariah', '22120765@student.ciu.edu.tr',
'+2347061614885', '2025-10-07');
```

```
INSERT INTO Member VALUES (2, 'Sultan', '22305640@student.ciu.edu.tr',
'+77052175021', '2025-09-07');

INSERT INTO Member VALUES (3, 'Josh', 'josh@student.ciu.edu.tr', '+9000000000', '2025-
08-01');

INSERT INTO Member VALUES (4, 'Joel', 'joel@student.ciu.edu.tr', '+904444444446', '2024-
07-15');

-- staff members

INSERT INTO Staff VALUES (1, 'Jacob Bush', 'Assistant');

INSERT INTO Staff VALUES (2, 'Aline Koj', 'Librarian');

INSERT INTO Staff VALUES (3, 'Mark Koop', 'Supervisor');

INSERT INTO Staff VALUES (4, 'Jil Umar', 'Manager');


-- Borrow

INSERT INTO Borrow VALUES (1, 1, 1, 1, '2025-03-01', '2025-03-15', '2025-03-20');

INSERT INTO Borrow VALUES (2, 2, 2, 2, '2025-03-05', '2025-03-19', '2025-03-18');

INSERT INTO Borrow VALUES (3, 3, 3, 3, '2025-03-10', '2025-03-24', NULL);

INSERT INTO Borrow VALUES (4, 4, 4, 4, '2025-03-12', '2025-03-26', '2025-03-30');

INSERT INTO Borrow VALUES (5, 1, 6, 2, '2025-02-01', '2025-02-15', '2025-02-14');


-- Fines

INSERT INTO Fine VALUES (1, 1, 5.00, 'UNPAID');

INSERT INTO Fine VALUES (2, 4, 7.00, 'PAID');

INSERT INTO Fine VALUES (3, 3, 3.50, 'UNPAID');
```

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;
```



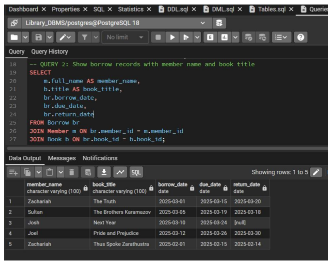
```
-- QUERY 1: Show books with their authors
SELECT
    b.title AS book_title,
    a.first_name,
    a.last_name
FROM Book b
JOIN Book_Author ba ON b.book_id = ba.book_id
JOIN Author a ON ba.author_id = a.author_id;
```

Showing rows: 1 to 7

	book_title character varying (100)	first_name character varying (50)	last_name character varying (50)
1	The Truth	Tresor	Kalombo
2	Next Year	Tresor	Kalombo
3	The Brothers Karamazov	Fyodor	Dostoevsky
4	Thus Spoke Zarathustra	Friedrich	Nietzsche
5	Pride and Prejudice	Jane	Austen
6	The Stranger	Albert	Camus
7	Notes from Underground	Fyodor	Dostoevsky

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;



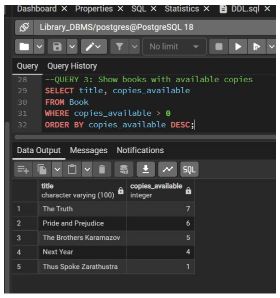
```
-- QUERY 2: Show borrow records with member name and book title
SELECT
    m.full_name AS member_name,
    b.title AS book_title,
    br.borrow_date,
    br.due_date,
    br.return_date
FROM Borrow br
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id;
```

Showing rows: 1 to 5

	member_name character varying (100)	book_title character varying (100)	borrow_date date	due_date date	return_date date
1	Zachariah	The Truth	2025-03-01	2025-03-15	2025-03-20
2	Sultan	The Brothers Karamazov	2025-03-05	2025-03-19	2025-03-18
3	Josh	Next Year	2025-03-10	2025-03-24	[null]
4	Joel	Pride and Prejudice	2025-03-12	2025-03-26	2025-03-30
5	Zachariah	Thus Spoke Zarathustra	2025-02-01	2025-02-15	2025-02-14

Query 3

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

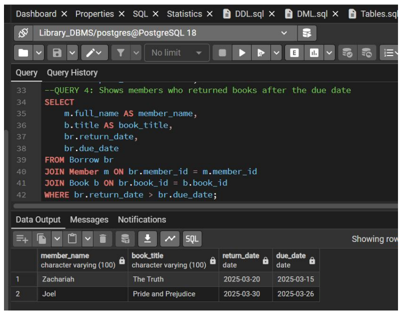


```
--QUERY 3: Show books with available copies
SELECT title, copies_available
FROM Book
WHERE copies_available > 0
ORDER BY copies_available DESC;
```

	title character varying (100)	copies_available integer
1	The Truth	7
2	Pride and Prejudice	6
3	The Brothers Karamazov	5
4	Next Year	4
5	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;

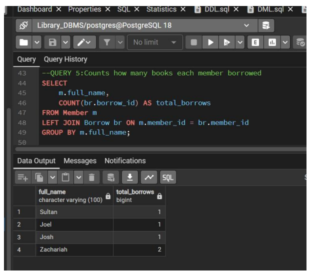


```sql
--QUERY 4: Shows members who returned books after the due date
SELECT
    m.full_name AS member_name,
    b.title AS book_title,
    br.return_date,
    br.due_date
FROM Borrow br
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id
WHERE br.return_date > br.due_date;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

	member_name character varying (100)	book_title character varying (100)	return_date date	due_date date
1	Zachariah	The Truth	2025-03-20	2025-03-15
2	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;

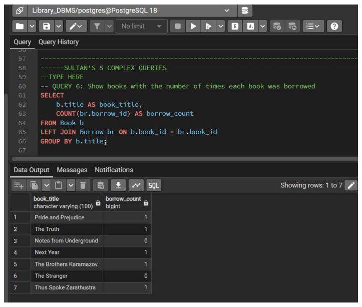


```sql
--QUERY 5:Counts how many books each member borrowed
SELECT
    m.full_name,
    COUNT(br.borrow_id) AS total_borrows
FROM Member m
LEFT JOIN Borrow br ON m.member_id = br.member_id
GROUP BY m.full_name;
```

	full_name character varying (100)	total_borrows bigint
1	Sultan	1
2	Joel	1
3	Josh	1
4	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;



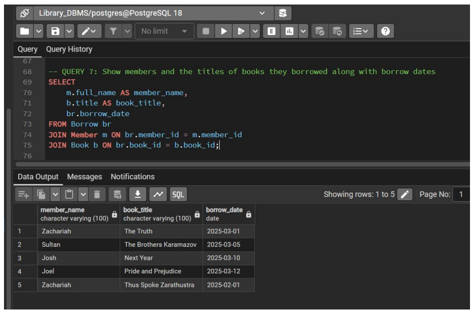
```sql
-----------------------------------------------------------------------
------SULTAN'S 5 COMPLEX QUERIES
--TYPE HERE
-- QUERY 6: Show books with the number of times each book was borrowed
SELECT
    b.title AS book_title,
    COUNT(br.borrow_id) AS borrow_count
FROM Book b
LEFT JOIN Borrow br ON b.book_id = br.book_id
GROUP BY b.title;
```

	book_title character varying (100)	borrow_count bigint
1	Pride and Prejudice	1
2	The Truth	1
3	Notes from Underground	0
4	Next Year	1
5	The Brothers Karamazov	1
6	The Stranger	0
7	Thus Spoke 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;



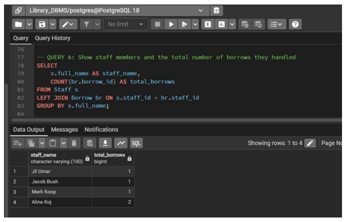
```sql
-- QUERY 7: Show members and the titles of books they borrowed along with borrow dates
SELECT
    m.full_name AS member_name,
    b.title AS book_title,
    br.borrow_date
FROM Borrow br
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id;
```

Showing rows: 1 to 5   Page No: 1

	member_name character varying (100)	book_title character varying (100)	borrow_date date
1	Zachariah	The Truth	2025-03-01
2	Sultan	The Brothers Karamazov	2025-03-05
3	Josh	Next Year	2025-03-10
4	Joel	Pride and Prejudice	2025-03-12
5	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;



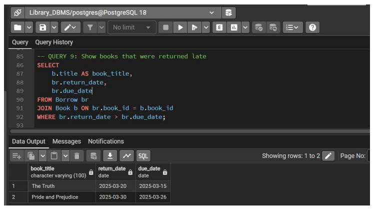
```sql
-- QUERY 8: Show staff members and the total number of borrows they handled
SELECT
    s.full_name AS staff_name,
    COUNT(br.borrow_id) AS total_borrows
FROM Staff s
LEFT JOIN Borrow br ON s.staff_id = br.staff_id
GROUP BY s.full_name;
```

Showing rows: 1 to 4   Page No

	staff_name character varying (100)	total_borrows bigint
1	Jil Umar	1
2	Jacob Bush	1
3	Mark Koop	1
4	Aline Koj	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;



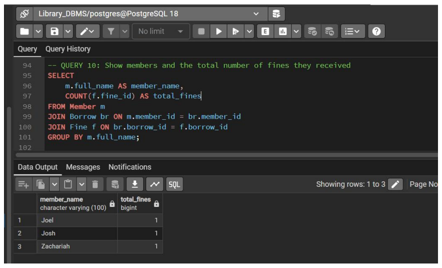
```sql
-- QUERY 9: Show books that were returned late
SELECT
    b.title AS book_title,
    br.return_date,
    br.due_date
FROM Borrow br
JOIN Book b ON br.book_id = b.book_id
WHERE br.return_date > br.due_date;
```

Showing rows: 1 to 2   Page No:

	book_title character varying (100)	return_date date	due_date date
1	The Truth	2025-03-20	2025-03-15
2	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;

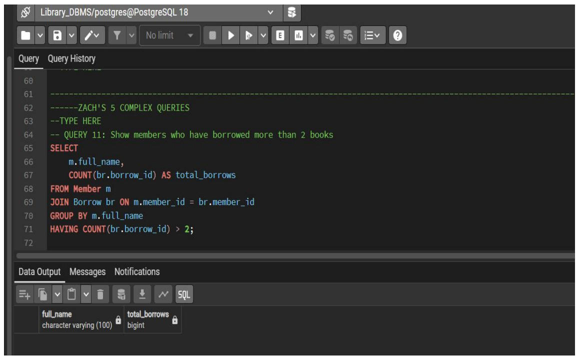


```
-- QUERY 10: Show members and the total number of fines they received
SELECT
    m.full_name AS member_name,
    COUNT(f.fine_id) AS total_fines
FROM Member m
JOIN Borrow br ON m.member_id = br.member_id
JOIN Fine f ON br.borrow_id = f.borrow_id
GROUP BY m.full_name;
```

	member_name character varying (100)	total_fines bigint
1	Joel	1
2	Josh	1
3	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;

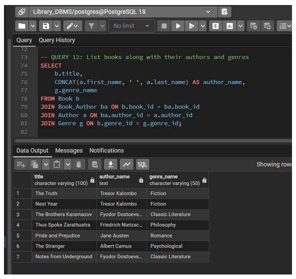


```
-------------------------------------------------------------------------------------------------------
------ZACH'S 5 COMPLEX QUERIES
--TYPE HERE
-- QUERY 11: Show members who have borrowed more than 2 books
SELECT
    m.full_name,
    COUNT(br.borrow_id) AS total_borrows
FROM Member m
JOIN Borrow br ON m.member_id = br.member_id
GROUP BY m.full_name
HAVING COUNT(br.borrow_id) > 2;
```

full_name character varying (100)	total_borrows bigint

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;

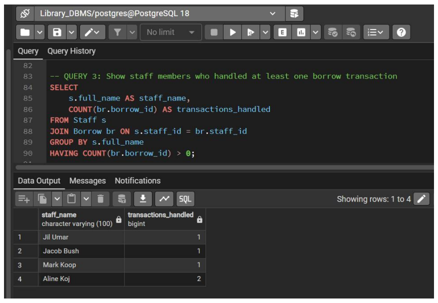


```
-- QUERY 12: List books along with their authors and genres
SELECT
    b.title,
    CONCAT(a.first_name, ' ', a.last_name) AS author_name,
    g.genre_name
FROM Book b
JOIN Book_Author ba ON b.book_id = ba.book_id
JOIN Author a ON ba.author_id = a.author_id
JOIN Genre g ON b.genre_id = g.genre_id;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

	title character varying (100)	author_name text	genre_name character varying (50)
1	The Truth	Tresor Kalombo	Fiction
2	Next Year	Tresor Kalombo	Fiction
3	The Brothers Karamazov	Fyodor Dostoevs...	Classic Literature
4	Thus Spoke Zarathustra	Friedrich Nietzsc...	Philosophy
5	Pride and Prejudice	Jane Austen	Romance
6	The Stranger	Albert Camus	Psychological
7	Notes from Underground	Fyodor Dostoevs...	Classic 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;


Library_DBMS/postgres@PostgreSQL 18

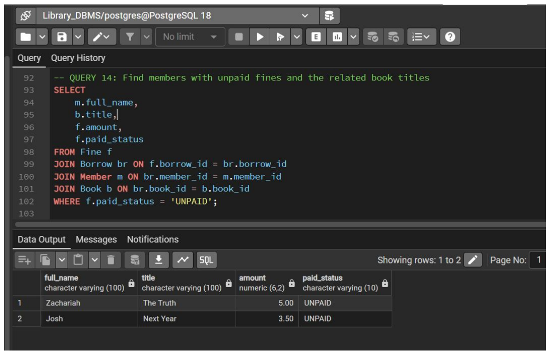
```sql
-- QUERY 3: Show staff members who handled at least one borrow transaction
SELECT
    s.full_name AS staff_name,
    COUNT(br.borrow_id) AS transactions_handled
FROM Staff s
JOIN Borrow br ON s.staff_id = br.staff_id
GROUP BY s.full_name
HAVING COUNT(br.borrow_id) > 0;
```

Showing rows: 1 to 4

	staff_name character varying (100)	transactions_handled bigint
1	Jil Umar	1
2	Jacob Bush	1
3	Mark Koop	1
4	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';


Library_DBMS/postgres@PostgreSQL 18

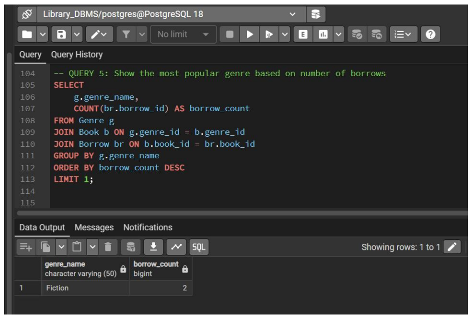
```sql
-- QUERY 14: Find members with unpaid fines and the related book titles
SELECT
    m.full_name,
    b.title,
    f.amount,
    f.paid_status
FROM Fine f
JOIN Borrow br ON f.borrow_id = br.borrow_id
JOIN Member m ON br.member_id = m.member_id
JOIN Book b ON br.book_id = b.book_id
WHERE f.paid_status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

Showing rows: 1 to 2 Page No: 1

	full_name character varying (100)	title character varying (100)	amount numeric (6,2)	paid_status character varying (10)
1	Zachariah	The Truth	5.00	UNPAID
2	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;


Library_DBMS/postgres@PostgreSQL 18

```sql
-- QUERY 5: 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;
```

Showing rows: 1 to 1

	genre_name character varying (50)	borrow_count bigint
1	Fiction	2