

LIBRARY DBMS

CMPE343 REPORT

0. Project Members and Contributions

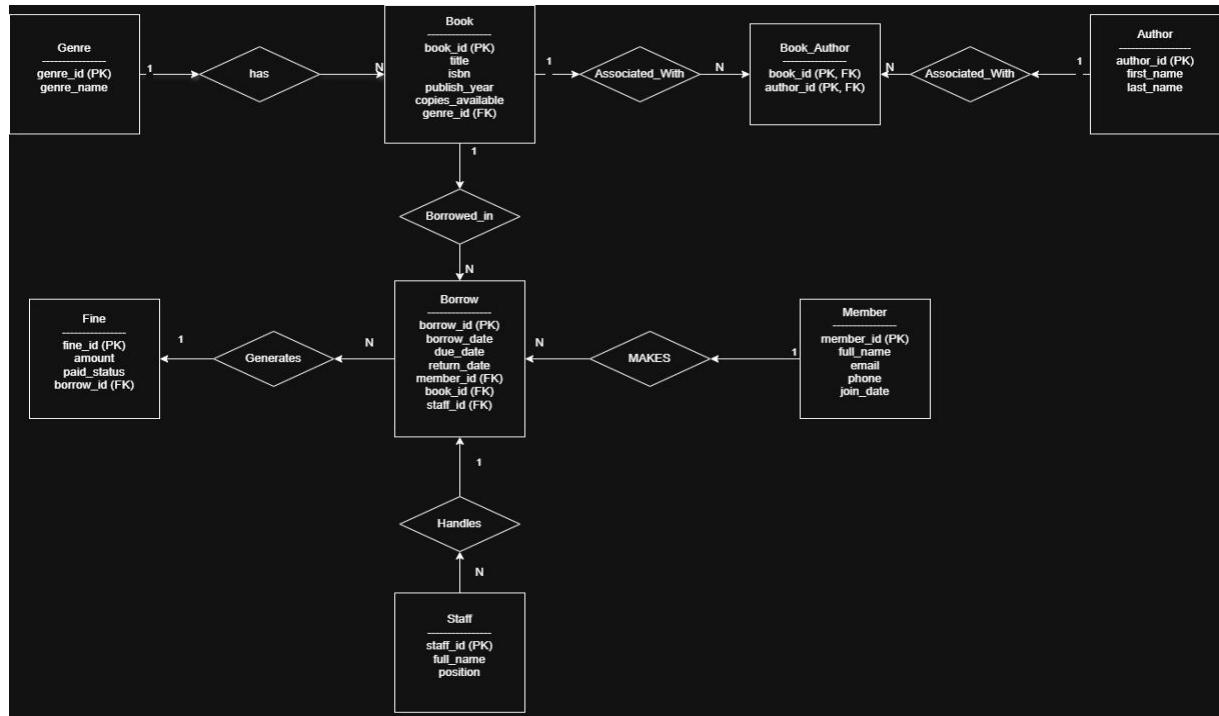
- TRESOR KALOMBO KALALA 22512651 : DDL, DML, QUERIES(1-5)
- Sultan Makhmut 22305640: ERD, DML, QUERIES(6-10)
- Dzaan Zachariah Allam 22120765: Report, DML, QUERIES(11-15)

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)**

-- TABLE 1: AUTHOR : Stores book authors

```
CREATE TABLE Author (
    author_id INT PRIMARY KEY, -----Unique author ID
    first_name VARCHAR(50) NOT NULL, -----Author first name
    last_name VARCHAR(50) NOT NULL -----Author last name
);
```

-- TABLE 2: BOOK: Stores books in the library

```
CREATE TABLE Book (
    book_id INT PRIMARY KEY, -----Unique book ID
    title VARCHAR(100) NOT NULL, -----Book title
    isbn VARCHAR(20) UNIQUE, -----ISBN number (unique)
    publish_year INT, -----Year of publication
    copies_available INT DEFAULT 1 -----Available copies
    CHECK (copies_available >= 0) -----Cannot be negative
);
```

-- TABLE 3: MEMBER: Stores library members

```
CREATE TABLE Member (
    member_id INT PRIMARY KEY, -----Unique member ID
    full_name VARCHAR(100) NOT NULL, -----Member name
    email VARCHAR(100) UNIQUE, -----Email (unique)
    phone VARCHAR(20), -----Phone number
    join_date DATE DEFAULT CURRENT_DATE -----Date joined
);
```

--TABLE 4: STAFF: Stores library staff

```
CREATE TABLE Staff (
    staff_id INT PRIMARY KEY, -----Unique staff ID
    full_name VARCHAR(100) NOT NULL, -----Staff name
    position VARCHAR(50) -- Job position
);
```

-- TABLE 5: BOOK_AUTHOR : Connects books and authors (many-to-many)

```
CREATE TABLE Book_Author (
    book_id INT, -----Book ID
    author_id INT, -----Author ID
    PRIMARY KEY (book_id, author_id), -----Composite primary key
    FOREIGN KEY (book_id) REFERENCES Book(book_id),
    FOREIGN KEY (author_id) REFERENCES Author(author_id)
);
```

--TABLE 6: BORROW: Stores borrow transactions

```
CREATE TABLE Borrow (
    borrow_id INT PRIMARY KEY, -----Borrow ID
    member_id INT, -----Member borrowing
    book_id INT, -----Borrowed book
    staff_id INT, -----Staff handling
    borrow_date DATE NOT NULL, -----Borrow date
    due_date DATE NOT NULL, -----Due date
    return_date DATE, -----Return 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)  
);
```

--TABLE 7: FINE : Stores fines for late returns

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

-- TABLE 8: GENRE : Stores book genres

```
CREATE TABLE Genre (  
genre_id INT PRIMARY KEY, -- Unique genre ID  
genre_name VARCHAR(50) NOT NULL -- Genre name  
);
```

● Data Manipulation Language (DML)

-- Authors

```
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, '2025-03-01', '2025-03-15', '2025-03-20');

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

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

INSERT INTO Borrow VALUES (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

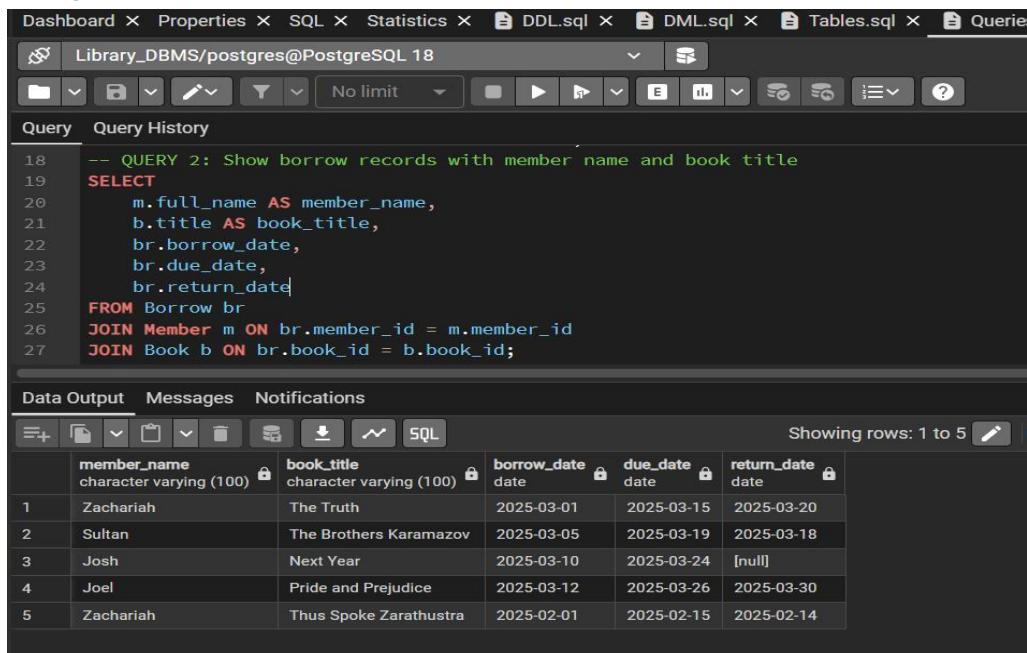
INSERT INTO Fine VALUES (1, 1, 5.00, 'UNPAID');

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

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

3. Queries

● Query 1



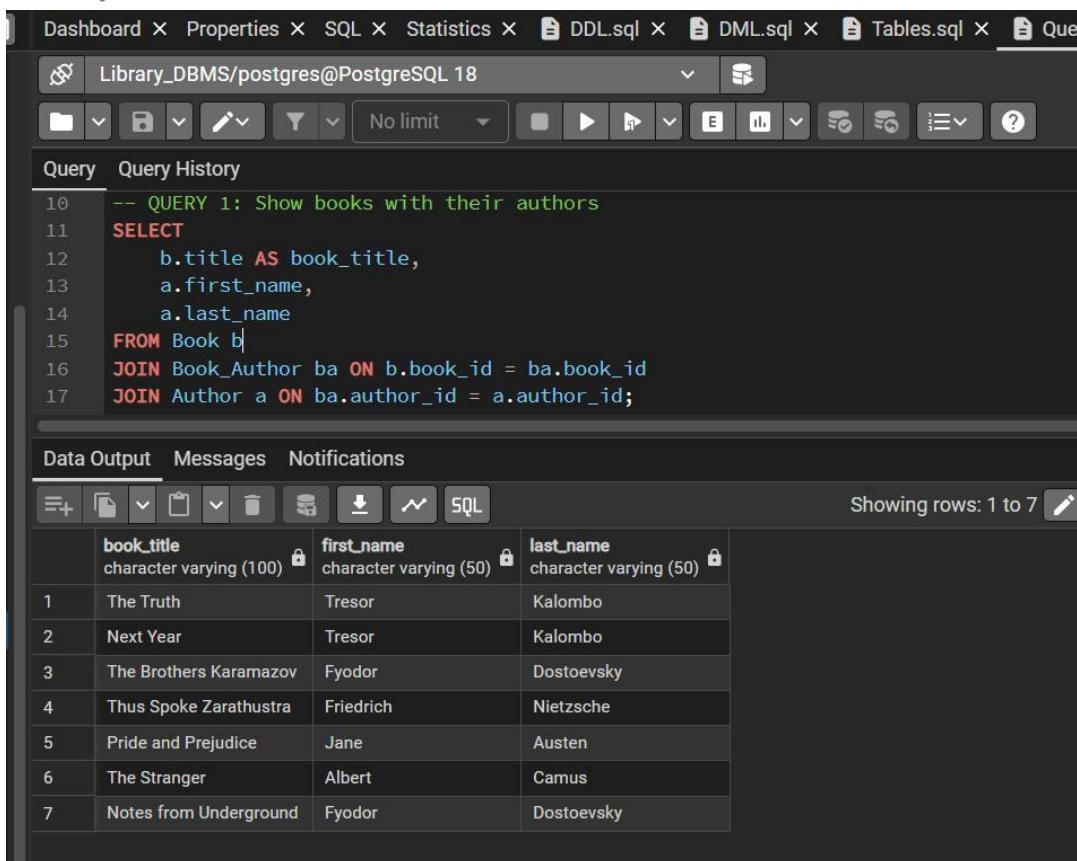
The screenshot shows the DBeaver interface with the SQL tab selected. The query window contains the following code:

```
-- 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;
```

The results pane shows the following data:

	member_name	book_title	borrow_date	due_date	return_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 2



The screenshot shows the DBeaver interface with the SQL tab selected. The query window contains the following code:

```
-- 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;
```

The results pane shows the following data:

	book_title	first_name	last_name
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 3

The screenshot shows the pgAdmin 4 interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The query editor contains the following SQL code:

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

The results pane displays the following table:

	title	copies_available
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

The screenshot shows the pgAdmin 4 interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The query editor contains the following SQL code:

```
--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;
```

The results pane displays the following table:

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

Query 5

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, and two open files: DDL.sql and DML.sql.
- Connection:** Library_DBMS/postgres@PostgreSQL 18.
- Query Editor:** Shows the following SQL code:

```
43 --QUERY 5:Counts how many books each member borrowed
44 SELECT
45     m.full_name,
46     COUNT(br.borrow_id) AS total_borrows
47 FROM Member m
48 LEFT JOIN Borrow br ON m.member_id = br.member_id
49 GROUP BY m.full_name;
```
- Data Output:** Shows the results of the query in a table:

	full_name	total_borrows
1	Sultan	1
2	Joel	1
3	Josh	1
4	Zachariah	2

Query 6

The screenshot shows the pgAdmin 4 interface with the following details:

- Toolbar:** Includes tabs for Dashboard, Properties, SQL, Statistics, and two open files: DDL.sql and DML.sql.
- Connection:** Library_DBMS/postgres@PostgreSQL 18.
- Query Editor:** Shows the following SQL code:

```
56 -----
57 -----SULTAN'S 5 COMPLEX QUERIES
58 --TYPE HERE
59 -- QUERY 6: Show books with the number of times each book was borrowed
60
61 SELECT
62     b.title AS book_title,
63     COUNT(br.borrow_id) AS borrow_count
64 FROM Book b
65 LEFT JOIN Borrow br ON b.book_id = br.book_id
66 GROUP BY b.title;
```
- Data Output:** Shows the results of the query in a table:

	book_title	borrow_count
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

The screenshot shows the pgAdmin 4 interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The query editor contains the following SQL code:

```
67
68 -- QUERY 7: Show members and the titles of books they borrowed along with borrow dates
69
70     m.full_name AS member_name,
71     b.title AS book_title,
72     br.borrow_date
73
74 FROM Borrow br
75 JOIN Member m ON br.member_id = m.member_id
76 JOIN Book b ON br.book_id = b.book_id;
```

The results pane shows the following data:

	member_name	book_title	borrow_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

The screenshot shows the pgAdmin 4 interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The query editor contains the following SQL code:

```
76
77 -- QUERY 8: Show staff members and the total number of borrows they handled
78
79     s.full_name AS staff_name,
80     COUNT(br.borrow_id) AS total_borrows
81
82 FROM Staff s
83 LEFT JOIN Borrow br ON s.staff_id = br.staff_id
84 GROUP BY s.full_name;
```

The results pane shows the following data:

	staff_name	total_borrows
1	Jil Umar	1
2	Jacob Bush	1
3	Mark Koop	1
4	Aline Koj	2

Query 9

The screenshot shows the pgAdmin interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The toolbar includes icons for file operations, search, and various database management functions. The main area has tabs for "Query" and "Query History", with "Query" selected. The query text is:

```
85 -- QUERY 9: Show books that were returned late
86 SELECT
87     b.title AS book_title,
88     br.return_date,
89     br.due_date
90 FROM Borrow br
91 JOIN Book b ON br.book_id = b.book_id
92 WHERE br.return_date > br.due_date;
93
```

The "Data Output" tab is active, displaying the results of the query:

	book_title	return_date	due_date
1	The Truth	2025-03-20	2025-03-15
2	Pride and Prejudice	2025-03-30	2025-03-26

At the bottom right of the data output area, it says "Showing rows: 1 to 2" and "Page No: []".

Query 10

The screenshot shows the pgAdmin interface with the title bar "Library_DBMS/postgres@PostgreSQL 18". The toolbar is identical to the previous screenshot. The main area has tabs for "Query" and "Query History", with "Query" selected. The query text is:

```
94 -- QUERY 10: Show members and the total number of fines they received
95 SELECT
96     m.full_name AS member_name,
97     COUNT(f.fine_id) AS total_fines
98 FROM Member m
99 JOIN Borrow br ON m.member_id = br.member_id
100 JOIN Fine f ON br.borrow_id = f.borrow_id
101 GROUP BY m.full_name;
102
```

The "Data Output" tab is active, displaying the results of the query:

	member_name	total_fines
1	Joel	1
2	Josh	1
3	Zachariah	1

At the bottom right of the data output area, it says "Showing rows: 1 to 3" and "Page No: []".

Query 11

```
60
61
62 -----ZACH'S 5 COMPLEX QUERIES
63 --TYPE HERE
64 -- QUERY 11: Show members who have borrowed more than 2 books
65 SELECT
66     m.full_name,
67     COUNT(br.borrow_id) AS total_borrows
68 FROM Member m
69 JOIN Borrow br ON m.member_id = br.member_id
70 GROUP BY m.full_name
71 HAVING COUNT(br.borrow_id) > 2;
72
```

Data Output Messages Notifications

full_name	total_borrows
character varying (100)	bigint

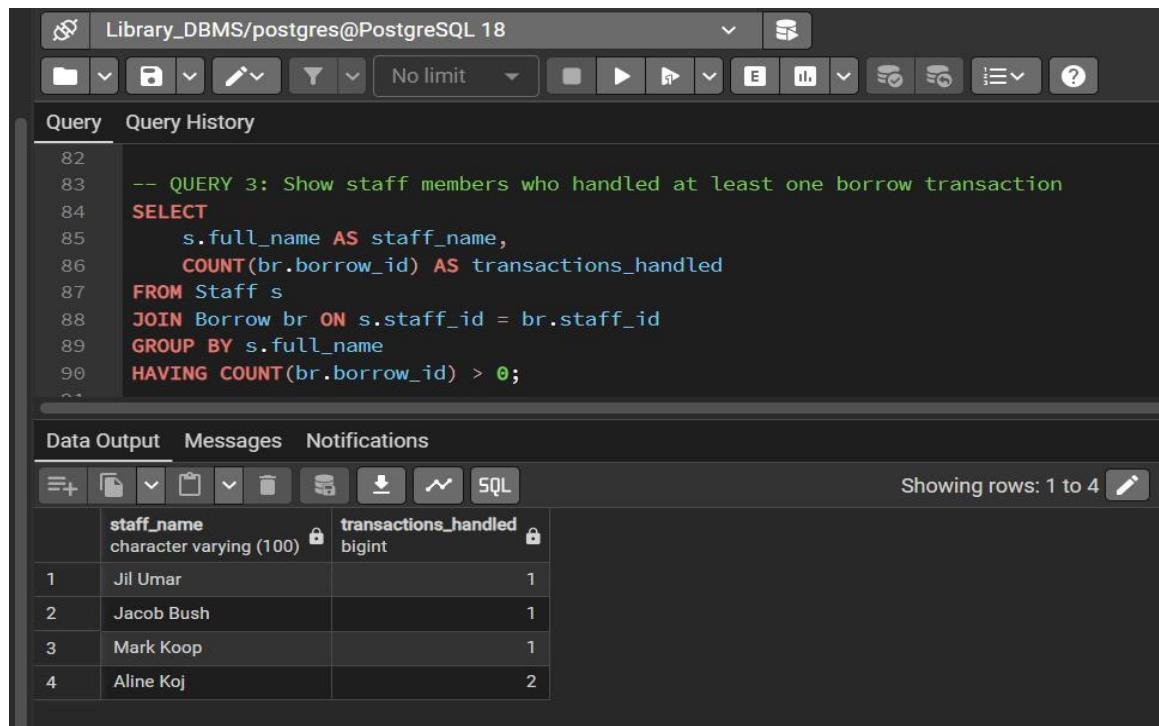
Query 12

```
72
73 -- QUERY 12: List books along with their authors and genres
74 SELECT
75     b.title,
76     CONCAT(a.first_name, ' ', a.last_name) AS author_name,
77     g.genre_name
78 FROM Book b
79 JOIN Book_Author ba ON b.book_id = ba.book_id
80 JOIN Author a ON ba.author_id = a.author_id
81 JOIN Genre g ON b.genre_id = g.genre_id;
82
```

Data Output Messages Notifications

	title	author_name	genre_name
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



The screenshot shows the pgAdmin 4 interface with a query editor and a data output viewer.

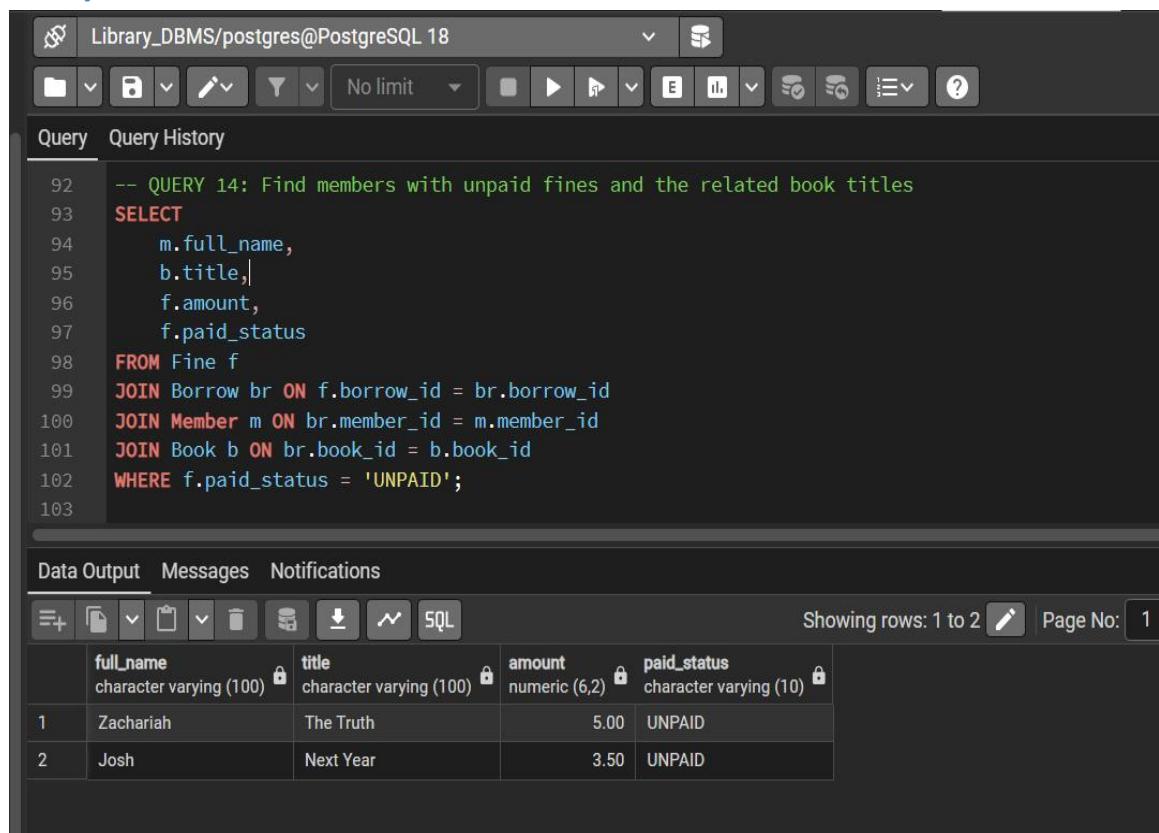
Query Editor:

```
82
83 -- QUERY 3: Show staff members who handled at least one borrow transaction
84
85     s.full_name AS staff_name,
86     COUNT(br.borrow_id) AS transactions_handled
87
88 FROM Staff s
89 JOIN Borrow br ON s.staff_id = br.staff_id
90 GROUP BY s.full_name
91 HAVING COUNT(br.borrow_id) > 0;
```

Data Output:

	staff_name	transactions_handled
1	Jil Umar	1
2	Jacob Bush	1
3	Mark Koop	1
4	Aline Koj	2

Query 14



The screenshot shows the pgAdmin 4 interface with a query editor and a data output viewer.

Query Editor:

```
92
93 -- QUERY 14: Find members with unpaid fines and the related book titles
94
95 SELECT
96     m.full_name,
97     b.title,
98     f.amount,
99     f.paid_status
100
101    FROM Fine f
102    JOIN Borrow br ON f.borrow_id = br.borrow_id
103    JOIN Member m ON br.member_id = m.member_id
104    JOIN Book b ON br.book_id = b.book_id
105    WHERE f.paid_status = 'UNPAID';
```

Data Output:

	full_name	title	amount	paid_status
1	Zachariah	The Truth	5.00	UNPAID
2	Josh	Next Year	3.50	UNPAID

Query 15

The screenshot shows a PostgreSQL query editor interface. The top bar displays the connection information: Library_DBMS/postgres@PostgreSQL 18. Below the connection bar is a toolbar with various icons for file operations, search, and navigation. The main area is divided into two tabs: 'Query' and 'Query History'. The 'Query' tab contains the following SQL code:

```
104 -- QUERY 5: Show the most popular genre based on number of borrows
105 SELECT
106     g.genre_name,
107     COUNT(br.borrow_id) AS borrow_count
108 FROM Genre g
109 JOIN Book b ON g.genre_id = b.genre_id
110 JOIN Borrow br ON b.book_id = br.book_id
111 GROUP BY g.genre_name
112 ORDER BY borrow_count DESC
113 LIMIT 1;
```

The 'Data Output' tab is selected, showing the results of the query. The results are presented in a table with two columns: 'genre_name' and 'borrow_count'. There is one row returned, showing 'Fiction' with a count of 2.

genre_name	borrow_count
character varying (50)	bigint
1 Fiction	2

Showing rows: 1 to 1