AI ASSISTED CODING

LAB-16: *Database Design and Queries: Schema Design and SQL Generation*

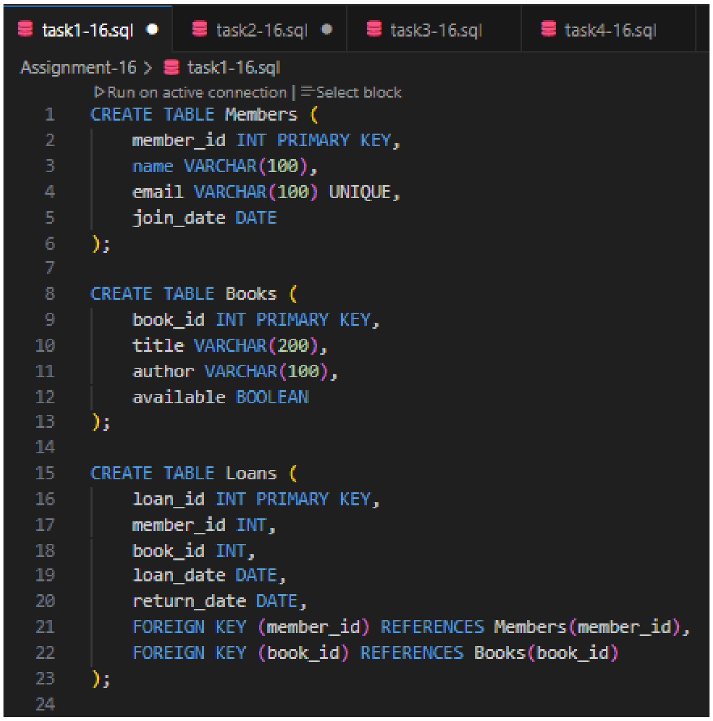
Roll no: 2503A51L44
Name: Meer Burhan Ali Hashmi
Batch:20

Task-1 Description: Schema Generation

Task: Ask AI to design a schema for a Library Management System (Tables: Books, Members, Loans).

Prompt: Design a schema for a Library Management System (Tables: Books, Members, Loans).

Code Generated:


task1-16.sql
task2-16.sql
task3-16.sql
task4-16.sql
Assignment-16 > task1-16.sql
▷ Run on active connection | ☰ Select block

```sql
CREATE TABLE Members (
    member_id INT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    join_date DATE
);

CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(200),
    author VARCHAR(100),
    available BOOLEAN
);

CREATE TABLE Loans (
    loan_id INT PRIMARY KEY,
    member_id INT,
    book_id INT,
    loan_date DATE,
    return_date DATE,
    FOREIGN KEY (member_id) REFERENCES Members(member_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

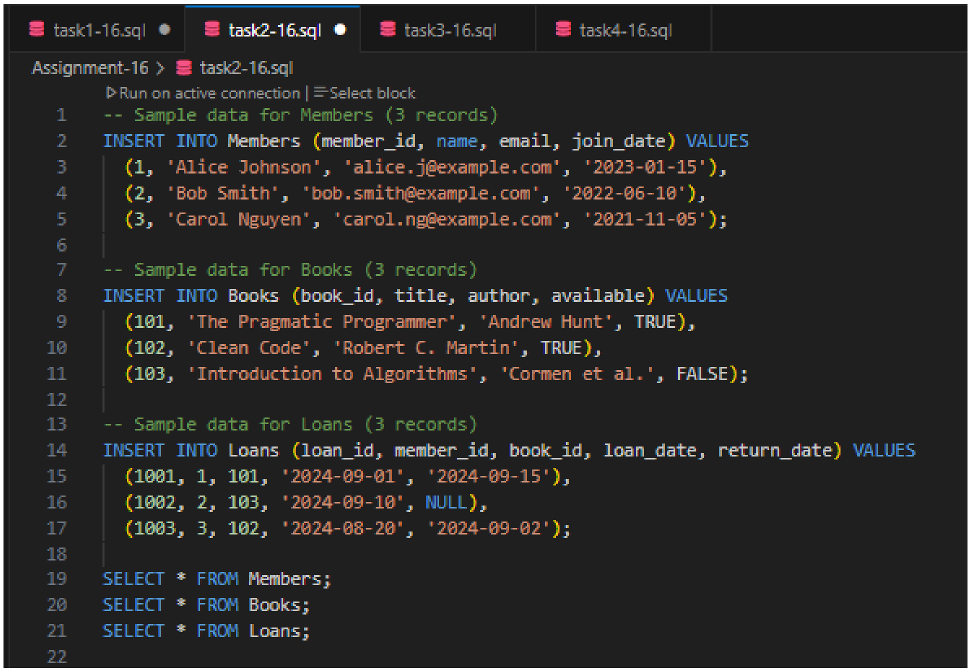
Observation: The AI successfully designed the Library Management System schema with the required tables and appropriate attributes to manage books, members, and loan details. It ensured proper structure that supports future query execution and relationships between tables. This task shows that AI can quickly and accurately create a complete database schema.

Task-2 Description: Error Insert Data

Task: Ask AI to generate INSERT INTO queries for the schema above (3 sample records per table).

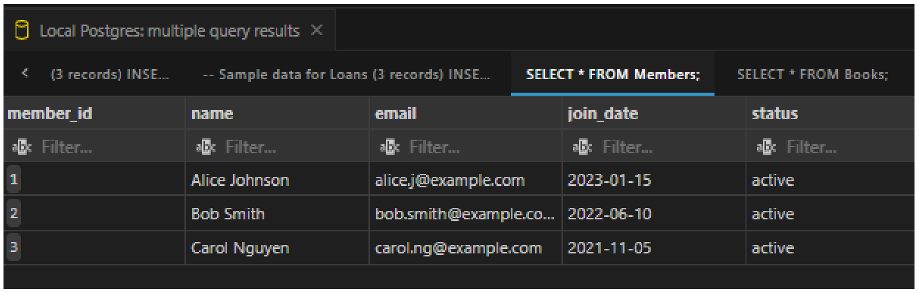
Prompt: Generate INSERT INTO queries for the schema above (3 sample records per table).

Code Generated:

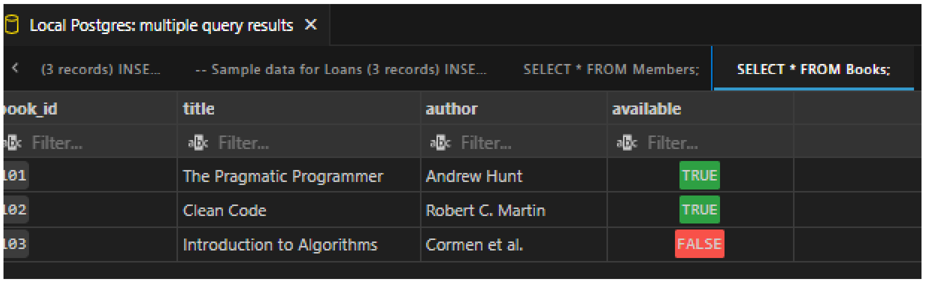

task1-16.sql
task2-16.sql
task3-16.sql
task4-16.sql
Assignment-16 > task2-16.sql
▷ Run on active connection | ☰ Select block

```sql
-- Sample data for Members (3 records)
INSERT INTO Members (member_id, name, email, join_date) VALUES
  (1, 'Alice Johnson', 'alice.j@example.com', '2023-01-15'),
  (2, 'Bob Smith', 'bob.smith@example.com', '2022-06-10'),
  (3, 'Carol Nguyen', 'carol.ng@example.com', '2021-11-05');

-- Sample data for Books (3 records)
INSERT INTO Books (book_id, title, author, available) VALUES
  (101, 'The Pragmatic Programmer', 'Andrew Hunt', TRUE),
  (102, 'Clean Code', 'Robert C. Martin', TRUE),
  (103, 'Introduction to Algorithms', 'Cormen et al.', FALSE);

-- Sample data for Loans (3 records)
INSERT INTO Loans (loan_id, member_id, book_id, loan_date, return_date) VALUES
  (1001, 1, 101, '2024-09-01', '2024-09-15'),
  (1002, 2, 103, '2024-09-10', NULL),
  (1003, 3, 102, '2024-08-20', '2024-09-02');

SELECT * FROM Members;
SELECT * FROM Books;
SELECT * FROM Loans;
```

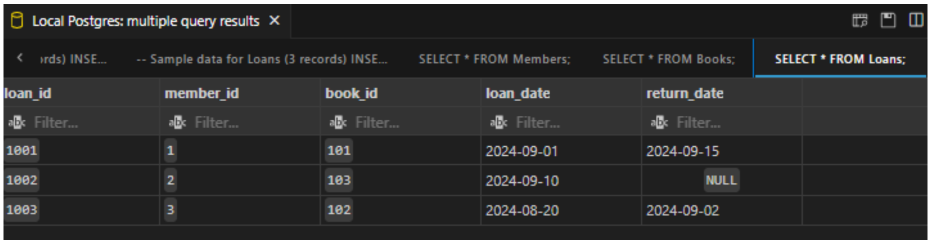
Output:


Local Postgres: multiple query results
(3 records) INSE...
-- Sample data for Loans (3 records) INSE...
SELECT * FROM Members;
SELECT * FROM Books;
Filter...

member_id	name	email	join_date	status
1	Alice Johnson	alice.j@example.com	2023-01-15	active
2	Bob Smith	bob.smith@example.co...	2022-06-10	active
3	Carol Nguyen	carol.ng@example.com	2021-11-05	active


Local Postgres: multiple query results
(3 records) INSE...
-- Sample data for Loans (3 records) INSE...
SELECT * FROM Members;
SELECT * FROM Books;
Filter...

ook_id	title	author	available
01	The Pragmatic Programmer	Andrew Hunt	TRUE
02	Clean Code	Robert C. Martin	TRUE
03	Introduction to Algorithms	Cormen et al.	FALSE



loan_id	member_id	book_id	loan_date	return_date
1001	1	101	2024-09-01	2024-09-15
1002	2	103	2024-09-10	NULL
1003	3	102	2024-08-20	2024-09-02

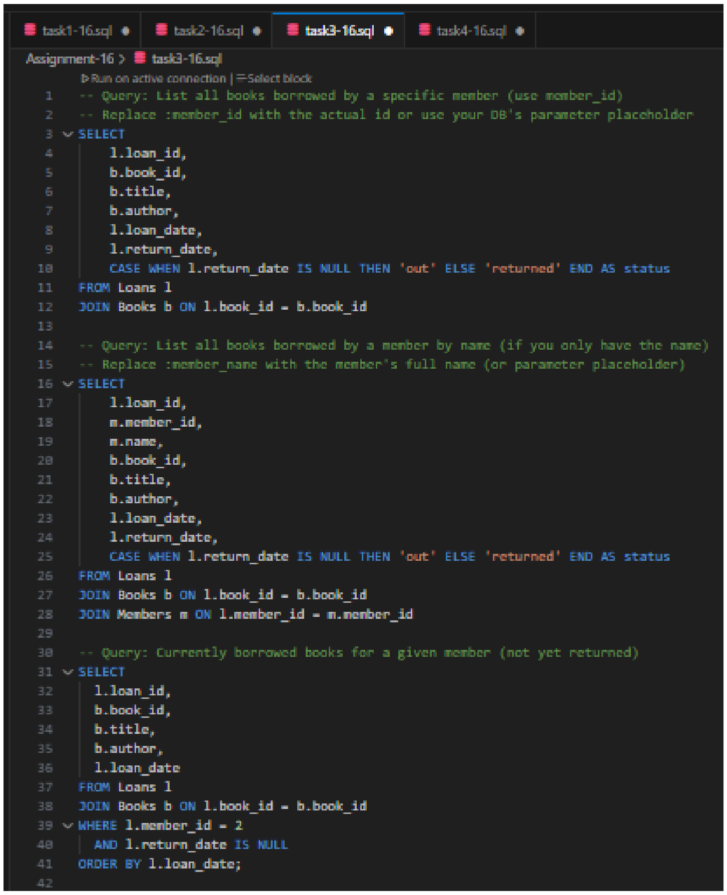
Observation:AI generated correct INSERT INTO statements with sample data for each of the tables created in Task-1. The execution of queries indicates proper syntax and ensures that the database has relevant test records. This task highlights AI’s usefulness in saving time while populating databases with meaningful sample data.

Task-3 Description:Basic Queries

Task: Use AI to generate a query to list all books borrowed by a specific member.

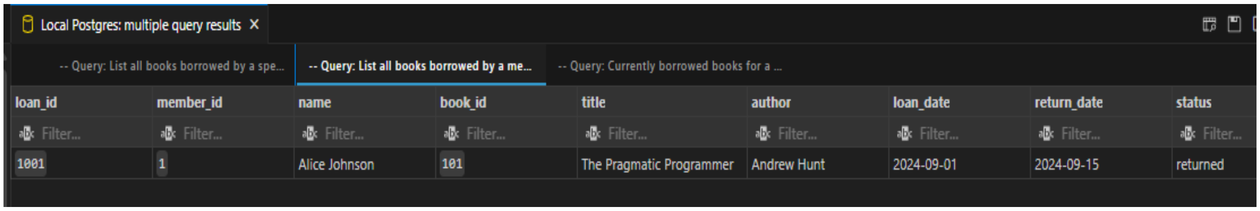
Prompt: Generate a query to list all books borrowed by a specific member.

Code Generated:

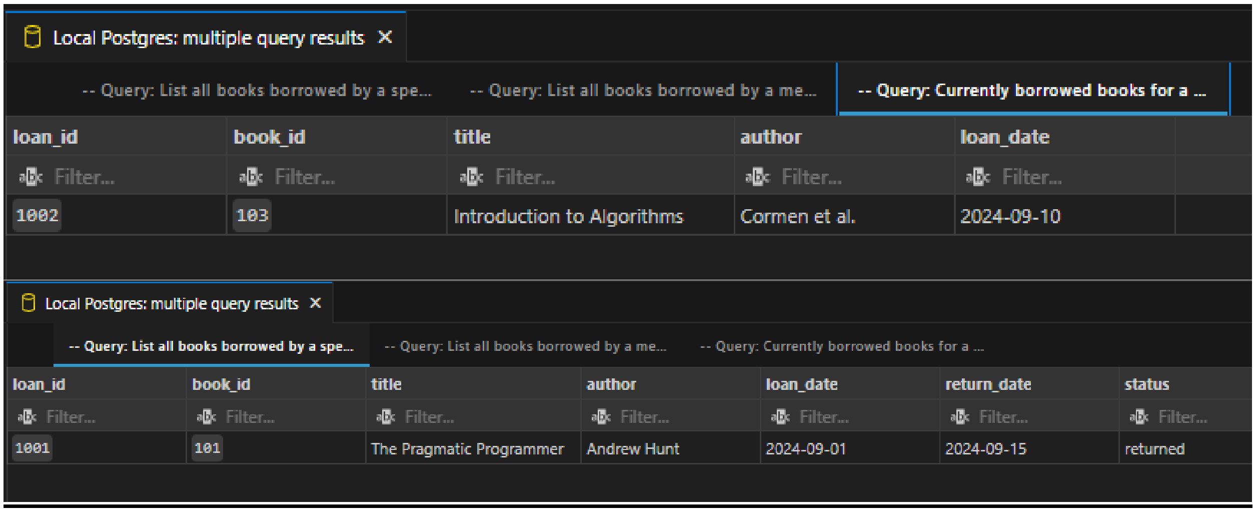

task1-16.sql
task2-16.sql
task3-16.sql
task4-16.sql
Assignment-16 > task3-16.sql
▷ Run on active connection | ☰ Select block

```sql
-- Query: List all books borrowed by a specific member (use member_id)
-- Replace :member_id with the actual id or use your DB's parameter placeholder
SELECT
    l.loan_id,
    b.book_id,
    b.title,
    b.author,
    l.loan_date,
    l.return_date,
    CASE WHEN l.return_date IS NULL THEN 'out' ELSE 'returned' END AS status
FROM Loans l
JOIN Books b ON l.book_id = b.book_id

-- Query: List all books borrowed by a member by name (if you only have the name)
-- Replace :member_name with the member's full name (or parameter placeholder)
SELECT
    l.loan_id,
    m.member_id,
    m.name,
    b.book_id,
    b.title,
    b.author,
    l.loan_date,
    l.return_date,
    CASE WHEN l.return_date IS NULL THEN 'out' ELSE 'returned' END AS status
FROM Loans l
JOIN Books b ON l.book_id = b.book_id
JOIN Members m ON l.member_id = m.member_id

-- Query: Currently borrowed books for a given member (not yet returned)
SELECT
  l.loan_id,
  b.book_id,
  b.title,
  b.author,
  l.loan_date
FROM Loans l
JOIN Books b ON l.book_id = b.book_id
WHERE l.member_id = 2
  AND l.return_date IS NULL
ORDER BY l.loan_date;
```

Output:


Local Postgres: multiple query results
-- Query: List all books borrowed by a spe...
-- Query: List all books borrowed by a me...
-- Query: Currently borrowed books for a ...
Filter...

loan_id	member_id	name	book_id	title	author	loan_date	return_date	status
1001	1	Alice Johnson	101	The Pragmatic Programmer	Andrew Hunt	2024-09-01	2024-09-15	returned


Local Postgres: multiple query results
-- Query: List all books borrowed by a spe...
-- Query: List all books borrowed by a me...
-- Query: Currently borrowed books for a ...
Filter...

loan_id	book_id	title	author	loan_date
1002	103	Introduction to Algorithms	Cormen et al.	2024-09-10

loan_id	book_id	title	author	loan_date	return_date	status
1001	101	The Pragmatic Programmer	Andrew Hunt	2024-09-01	2024-09-15	returned

Observation: The AI provided an accurate SQL SELECT query to list books borrowed by a specific member, using table relationships effectively. It ensures correct retrieval of data from multiple tables, showing the practical application of the designed schema. This demonstrates how AI helps in generating precise queries for real-time information access.

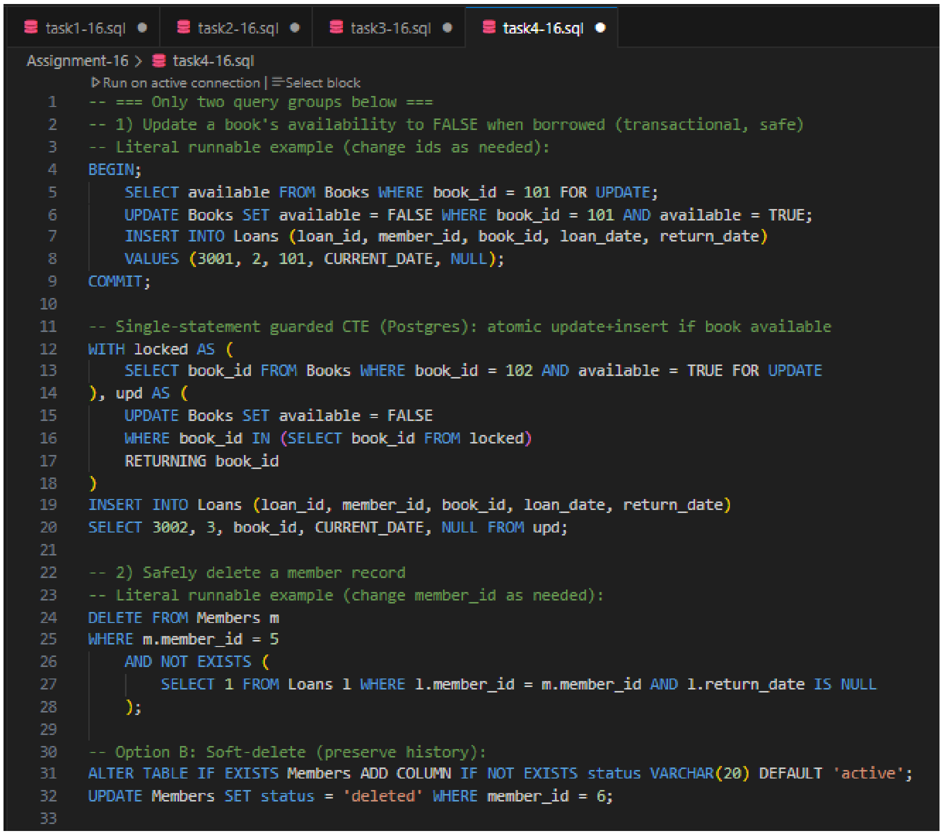
Task-4 Description: Update and Delete Queries

Task: Generate queries with AI for:

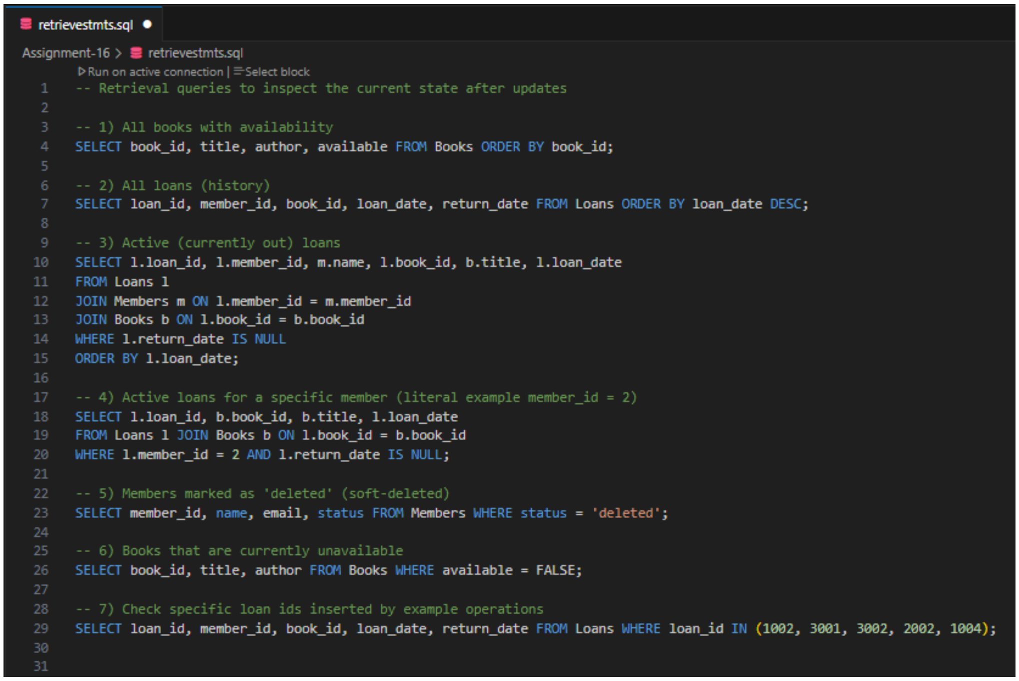
- Updating a book's availability to FALSE when borrowed.
- Deleting a member record safely.

Prompt: Generate queries for:

- Updating a book's availability to FALSE when borrowed.


task1-16.sql
task2-16.sql
task3-16.sql
task4-16.sql
Assignment-16 > task4-16.sql
Run on active connection | Select block

```sql
-- === Only two query groups below ===
-- 1) Update a book's availability to FALSE when borrowed (transactional, safe)
-- Literal runnable example (change ids as needed):
BEGIN;
    SELECT available FROM Books WHERE book_id = 101 FOR UPDATE;
    UPDATE Books SET available = FALSE WHERE book_id = 101 AND available = TRUE;
    INSERT INTO Loans (loan_id, member_id, book_id, loan_date, return_date)
    VALUES (3001, 2, 101, CURRENT_DATE, NULL);
COMMIT;

-- Single-statement guarded CTE (Postgres): atomic update+insert if book available
WITH locked AS (
    SELECT book_id FROM Books WHERE book_id = 102 AND available = TRUE FOR UPDATE
), upd AS (
    UPDATE Books SET available = FALSE
    WHERE book_id IN (SELECT book_id FROM locked)
    RETURNING book_id
)
INSERT INTO Loans (loan_id, member_id, book_id, loan_date, return_date)
SELECT 3002, 3, book_id, CURRENT_DATE, NULL FROM upd;

-- 2) Safely delete a member record
-- Literal runnable example (change member_id as needed):
DELETE FROM Members m
WHERE m.member_id = 5
    AND NOT EXISTS (
        SELECT 1 FROM Loans l WHERE l.member_id = m.member_id AND l.return_date IS NULL
    );

-- Option B: Soft-delete (preserve history):
ALTER TABLE IF EXISTS Members ADD COLUMN IF NOT EXISTS status VARCHAR(20) DEFAULT 'active';
UPDATE Members SET status = 'deleted' WHERE member_id = 6;

```

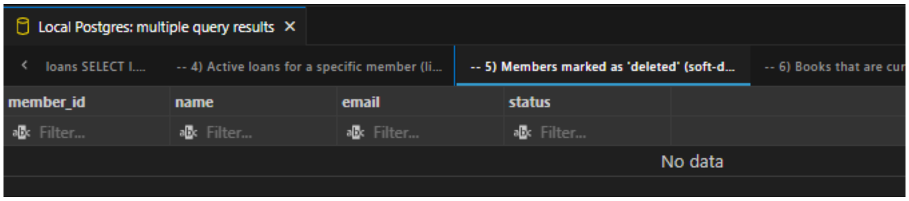

retrievestmts.sql
Assignment-16 > retrievestmts.sql
Run on active connection | Select block

```sql
-- Retrieval queries to inspect the current state after updates

-- 1) All books with availability
SELECT book_id, title, author, available FROM Books ORDER BY book_id;

-- 2) All loans (history)
SELECT loan_id, member_id, book_id, loan_date, return_date FROM Loans ORDER BY loan_date DESC;

-- 3) Active (currently out) loans
SELECT l.loan_id, l.member_id, m.name, l.book_id, b.title, l.loan_date
FROM Loans l
JOIN Members m ON l.member_id = m.member_id
JOIN Books b ON l.book_id = b.book_id
WHERE l.return_date IS NULL
ORDER BY l.loan_date;

-- 4) Active loans for a specific member (literal example member_id = 2)
SELECT l.loan_id, b.book_id, b.title, l.loan_date
FROM Loans l JOIN Books b ON l.book_id = b.book_id
WHERE l.member_id = 2 AND l.return_date IS NULL;

-- 5) Members marked as 'deleted' (soft-deleted)
SELECT member_id, name, email, status FROM Members WHERE status = 'deleted';

-- 6) Books that are currently unavailable
SELECT book_id, title, author FROM Books WHERE available = FALSE;

-- 7) Check specific loan ids inserted by example operations
SELECT loan_id, member_id, book_id, loan_date, return_date FROM Loans WHERE loan_id IN (1002, 3001, 3002, 2002, 1004);
```

- Deleting a member record safely.

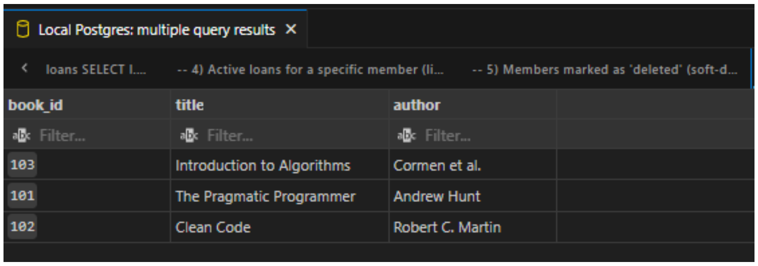
Code Generated:

Output:

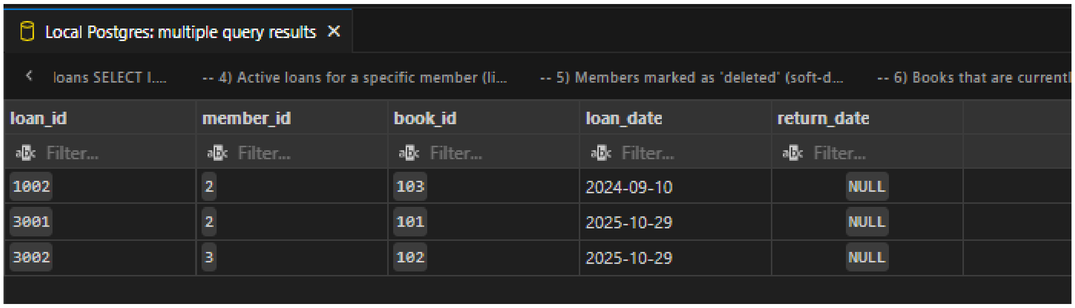

Local Postgres: multiple query results
loans SELECT l....
-- 4) Active loans for a specific member (li...
-- 5) Members marked as 'deleted' (soft-d...
-- 6) Books that are cur

member_id	name	email	status

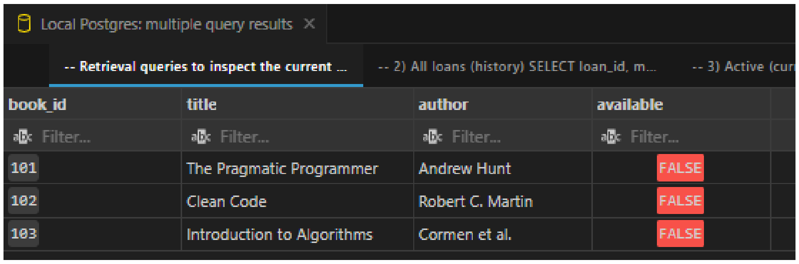
No data


Local Postgres: multiple query results
loans SELECT l....
-- 4) Active loans for a specific member (li...
-- 5) Members marked as 'deleted' (soft-d...

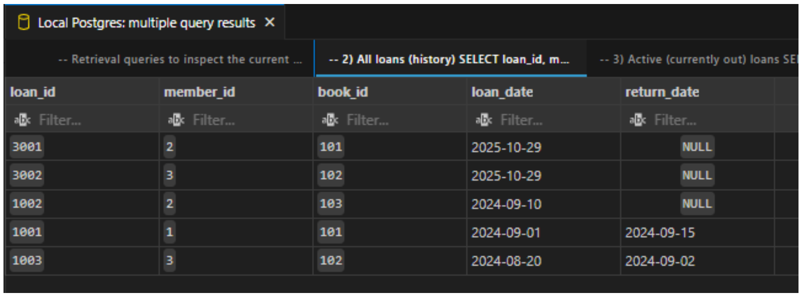
book_id	title	author
103	Introduction to Algorithms	Cormen et al.
101	The Pragmatic Programmer	Andrew Hunt
102	Clean Code	Robert C. Martin


Local Postgres: multiple query results
loans SELECT l....
-- 4) Active loans for a specific member (li...
-- 5) Members marked as 'deleted' (soft-d...
-- 6) Books that are current

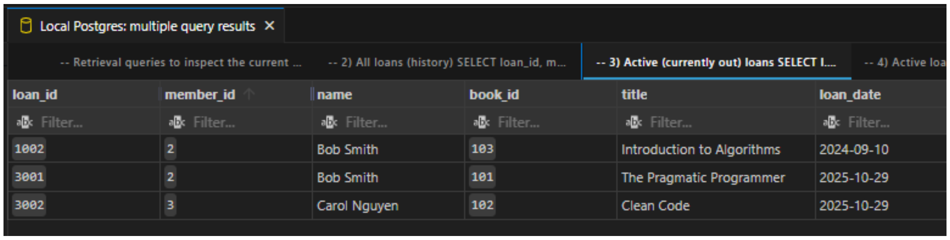
loan_id	member_id	book_id	loan_date	return_date
1002	2	103	2024-09-10	NULL
3001	2	101	2025-10-29	NULL
3002	3	102	2025-10-29	NULL


Local Postgres: multiple query results
-- Retrieval queries to inspect the current ...
-- 2) All loans (history) SELECT loan_id, m...
-- 3) Active (cur

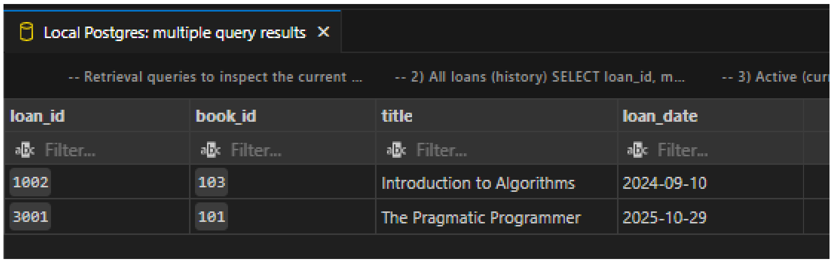
book_id	title	author	available
101	The Pragmatic Programmer	Andrew Hunt	FALSE
102	Clean Code	Robert C. Martin	FALSE
103	Introduction to Algorithms	Cormen et al.	FALSE


Local Postgres: multiple query results
-- Retrieval queries to inspect the current ...
-- 2) All loans (history) SELECT loan_id, m...
-- 3) Active (currently out) loans SE

loan_id	member_id	book_id	loan_date	return_date
3001	2	101	2025-10-29	NULL
3002	3	102	2025-10-29	NULL
1002	2	103	2024-09-10	NULL
1001	1	101	2024-09-01	2024-09-15
1003	3	102	2024-08-20	2024-09-02


Local Postgres: multiple query results
-- Retrieval queries to inspect the current ...
-- 2) All loans (history) SELECT loan_id, m...
-- 3) Active (currently out) loans SELECT l....
-- 4) Active loa

loan_id	member_id	name	book_id	title	loan_date
1002	2	Bob Smith	103	Introduction to Algorithms	2024-09-10
3001	2	Bob Smith	101	The Pragmatic Programmer	2025-10-29
3002	3	Carol Nguyen	102	Clean Code	2025-10-29


Local Postgres: multiple query results
-- Retrieval queries to inspect the current ...
-- 2) All loans (history) SELECT loan_id, m...
-- 3) Active (cur

loan_id	book_id	title	loan_date
1002	103	Introduction to Algorithms	2024-09-10
3001	101	The Pragmatic Programmer	2025-10-29

Observation:AI generated appropriate UPDATE and DELETE queries to manage changes in the database, including updating book availability and removing member details safely. The commands reflect secure handling of data modifications without affecting integrity. This task proves that AI can assist in maintaining and controlling database operations efficiently.