ASSIGNMENT – 16.4

NAME: SRIYA DARIPELLI

H.NO: 2403A51350

SUBJECT:AI ASSISTANT CODING

BATCH:01

Task 1: Create Database Schema Instructions:

* Using AI tools, generate CREATE TABLE statements for a

Library Database.

* Tables: Authors, Books, Members. Include primary keys, foreign keys, and appropriate data types.
* Ensure the design follows normalization rules

(1NF, 2NF, 3NF).

Starter Code Example for AI Completion:

-- Use AI to complete schema generation CREATE TABLE Authors ( author\_id INT PRIMARY KEY, name VARCHAR(100), country VARCHAR(50)

);

CREATE TABLE Books (

book\_id INT PRIMARY KEY, title VARCHAR(100), author\_id INT, published\_year INT,

FOREIGN KEY (author\_id) REFERENCES

Authors(author\_id)

);

CREATE TABLE Members ( member\_id INT PRIMARY KEY, name VARCHAR(100), email VARCHAR(100)

);

PROMPT:

develop a code to generate Explore how AI can optimize queries and ensure normalization. Task 1: Create Database Schema Instructions: • Using AI tools, generate CREATE TABLE statements for a Library Database. • Tables: Authors, Books, Members. Include primary keys, foreign keys, and appropriate data types. • Ensure the design follows normalization rules (1NF, 2NF, 3NF). Starter Code Example for AI Completion: -- Use AI to complete schema generation CREATE TABLE Authors ( author\_id INT PRIMARY KEY, name VARCHAR(100), country VARCHAR(50) ); CREATE TABLE Books ( book\_id INT PRIMARY KEY, title VARCHAR(100), author\_id INT, published\_year INT, FOREIGN KEY

(author\_id) REFERENCES Authors(author\_id) );

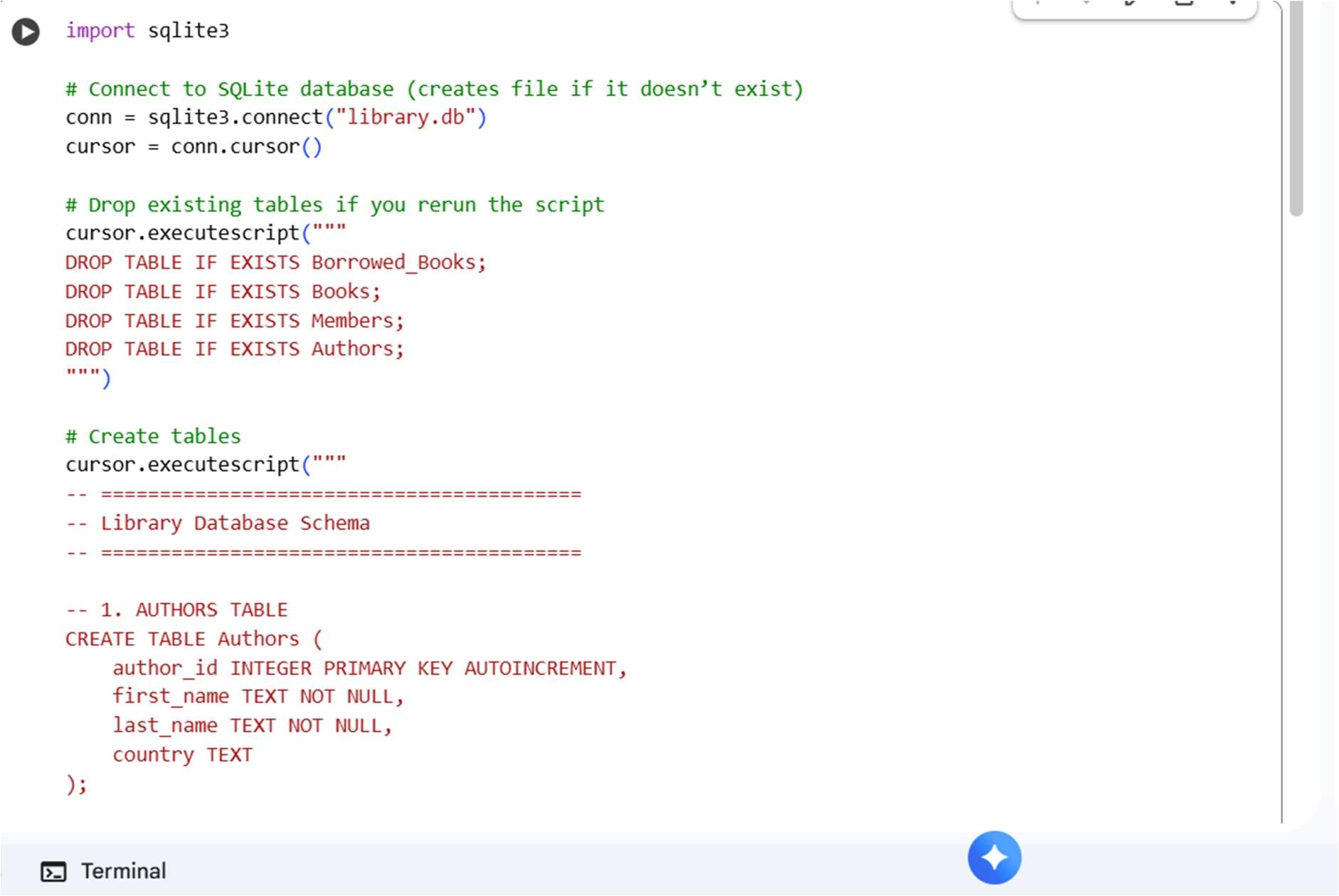
CREATE TABLE Members ( member\_id INT

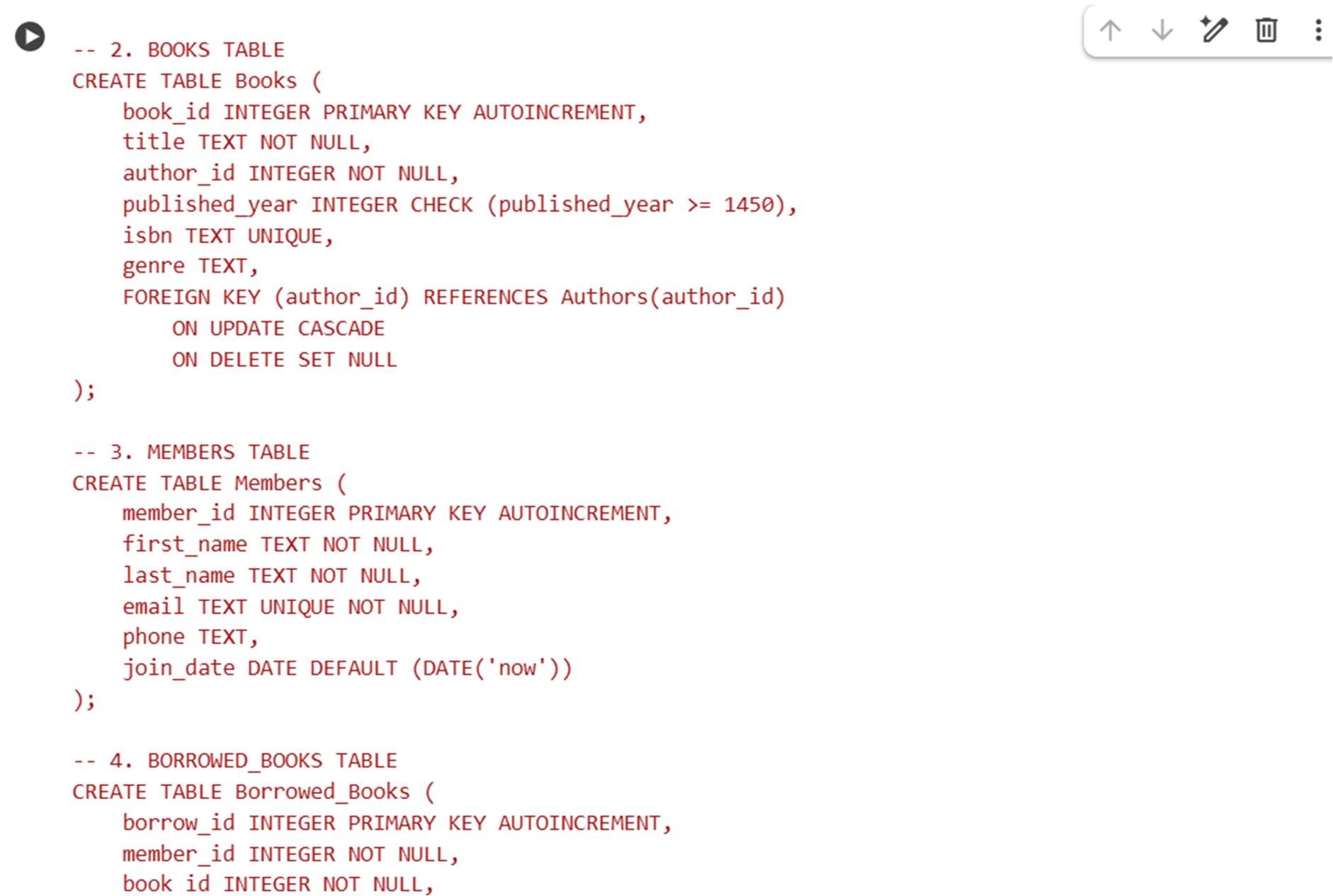
PRIMARY KEY, name VARCHAR(100), email

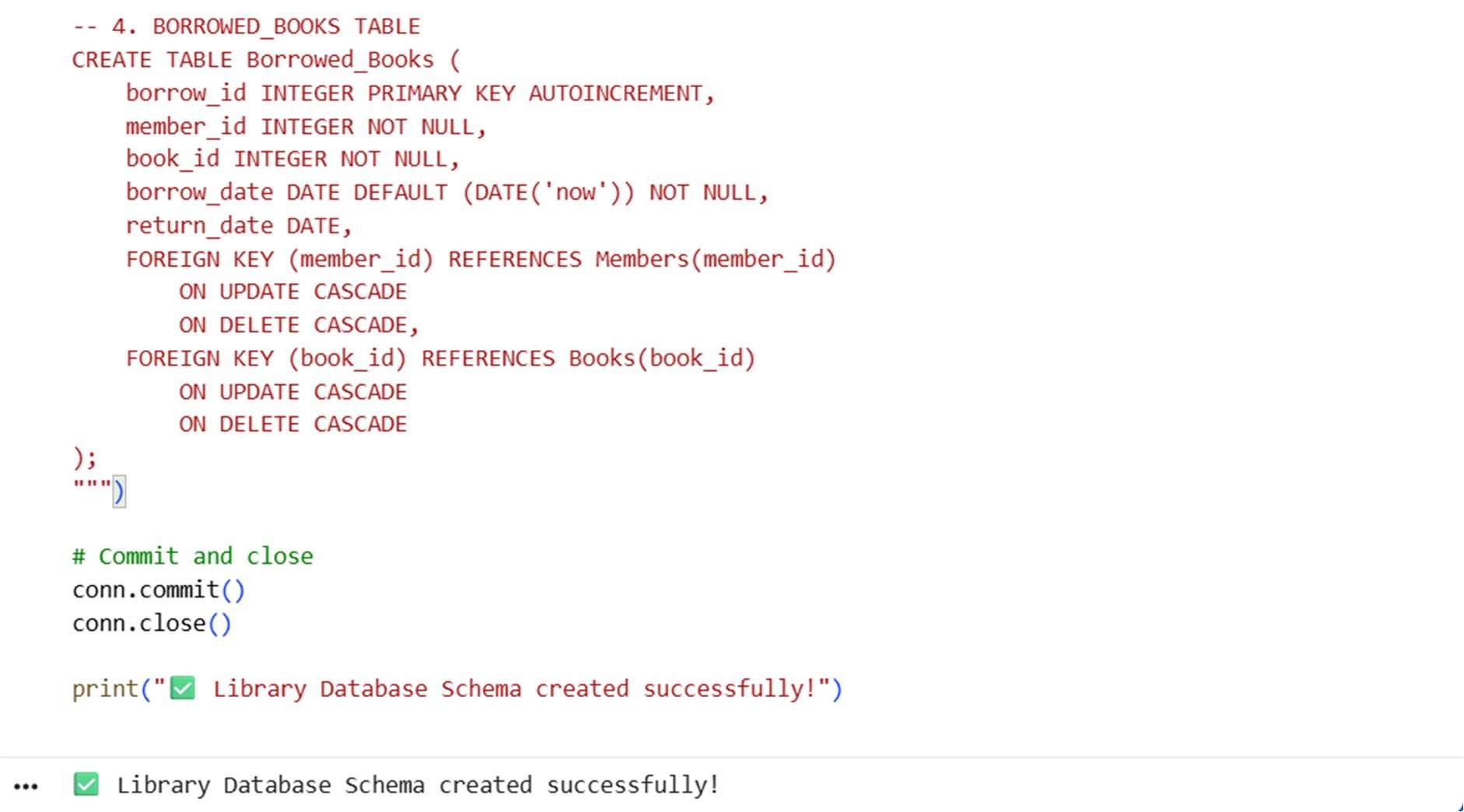
VARCHAR(100) );

Expected Output:

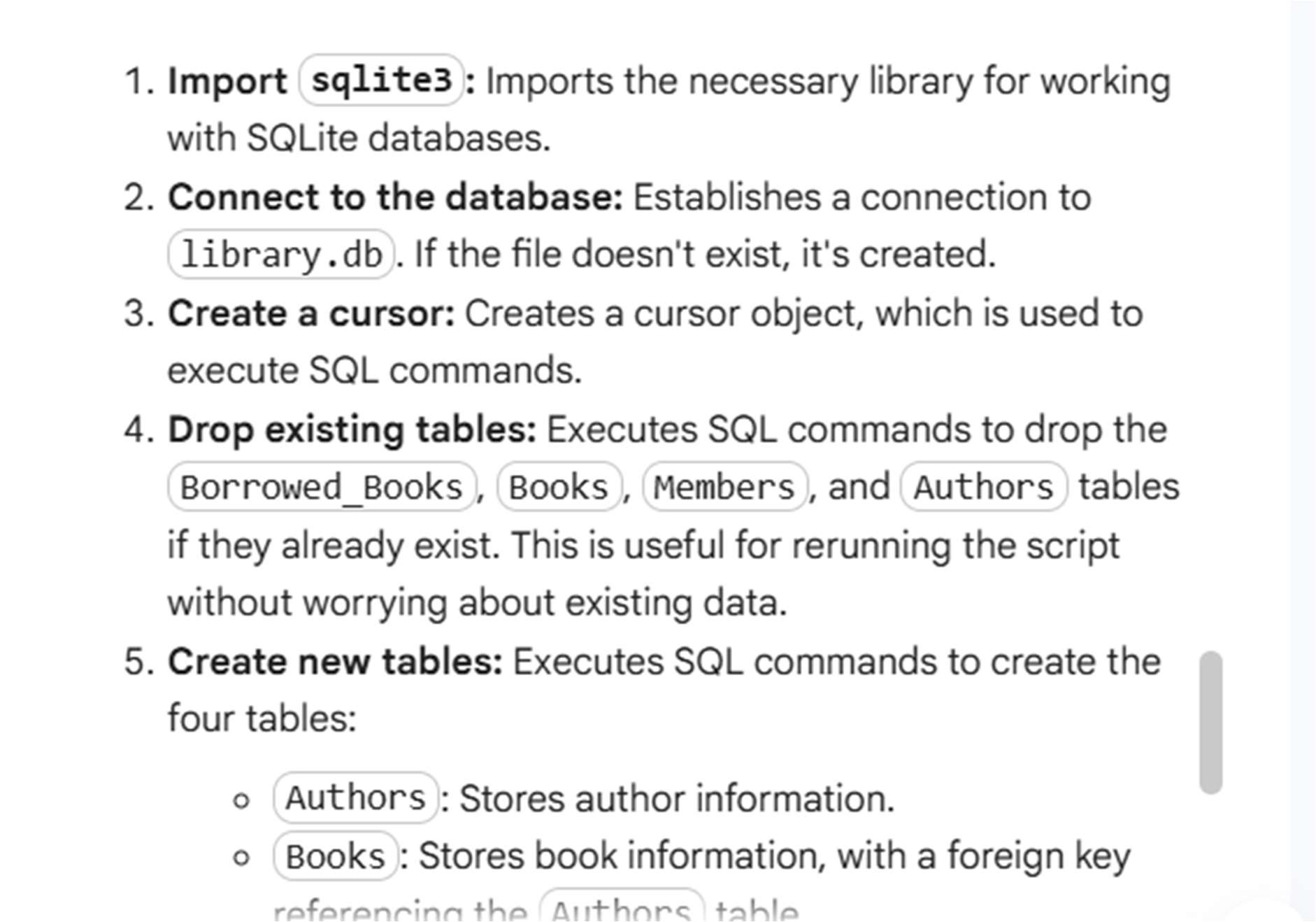
* Tables Authors, Books, Members are created successfully.
* Foreign key relationships are enforced

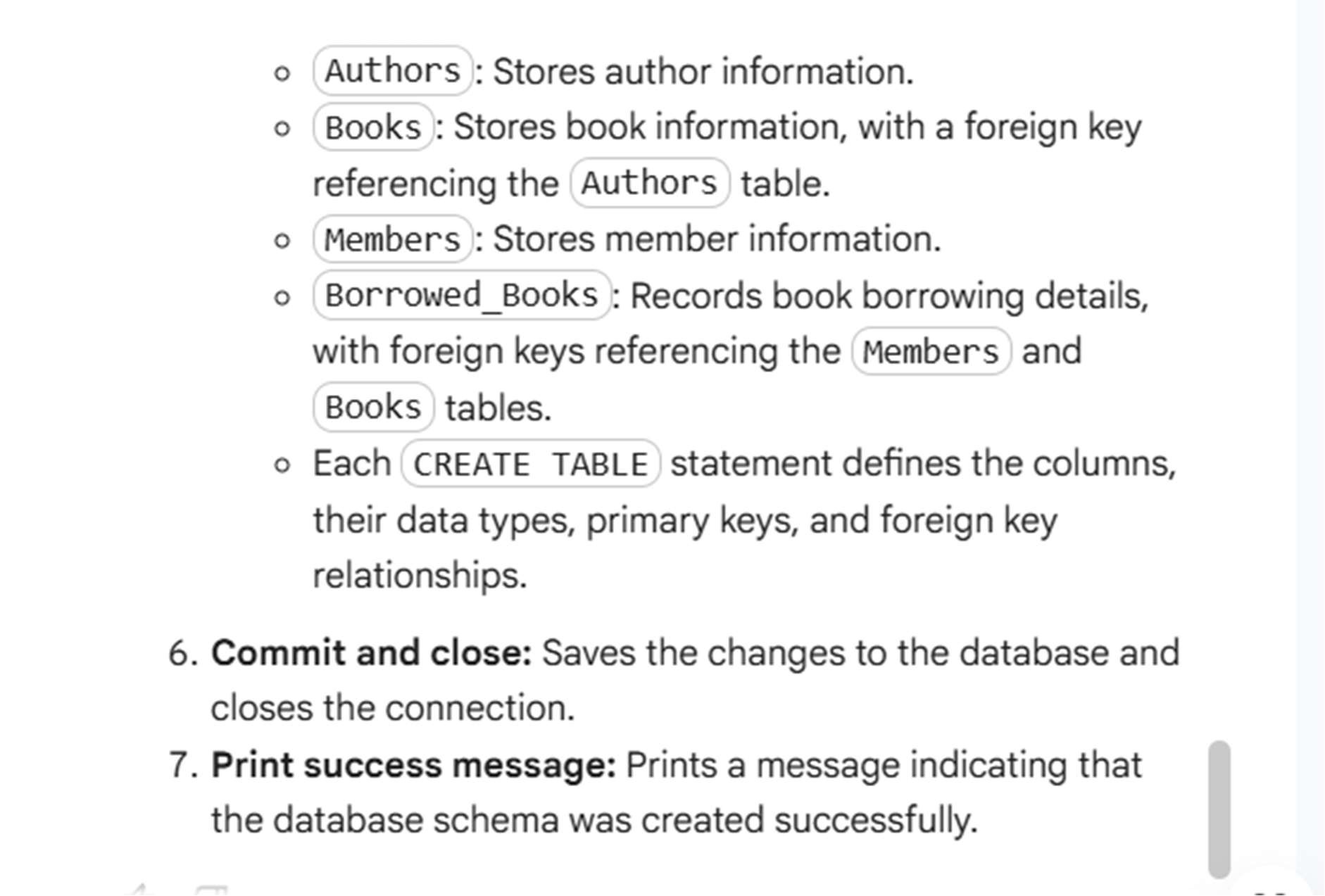






EXPLANATION:





TASK 2:

Use AI tools to generate INSERT statements to add sample data: o At least 3 authors, 5 books, and 3 members.

* Test the data by running SELECT statements.

Starter Code Example:

INSERT INTO Authors (author\_id, name, country)

VALUES

(1, 'J.K. Rowling', 'UK'),

(2, 'George R.R. Martin', 'USA'),

(3, 'Agatha Christie', 'UK');

INSERT INTO Books (book\_id, title, author\_id, published\_year)

VALUES

(1, 'Harry Potter', 1, 1997),

(2, 'A Game of Thrones', 2, 1996),

(3, 'Murder on the Orient Express', 3, 1934),

(4, 'Harry Potter 2', 1, 1998),

(5, 'A Clash of Kings', 2, 1998);

INSERT INTO Members (member\_id, name, email)

VALUES

(1, 'Alice', 'alice@example.com'),

(2, 'Bob', 'bob@example.com'), (3, 'Charlie', 'charlie@example.com'); Expected Output:

* Sample data inserted correctly.
* SELECT \* FROM Books; shows all 5 books.
* SELECT \* FROM Members; shows all 3 members.

PROMPT:

develop a python code to generate Use AI tools to generate INSERT statements to add sample data: o At least 3 authors, 5 books, and 3 members. • Test the data by running SELECT statements. Starter Code Example: INSERT INTO Authors (author\_id, name, country) VALUES (1, 'J.K. Rowling', 'UK'), (2, 'George R.R. Martin', 'USA'), (3, 'Agatha Christie', 'UK'); INSERT INTO Books (book\_id, title, author\_id, published\_year) VALUES (1, 'Harry Potter', 1, 1997), (2, 'A Game of Thrones', 2, 1996), (3, 'Murder on the

Orient Express', 3, 1934), (4, 'Harry Potter 2', 1,

1998), (5, 'A Clash of Kings', 2, 1998); INSERT INTO

Members (member\_id, name, email) VALUES (1,

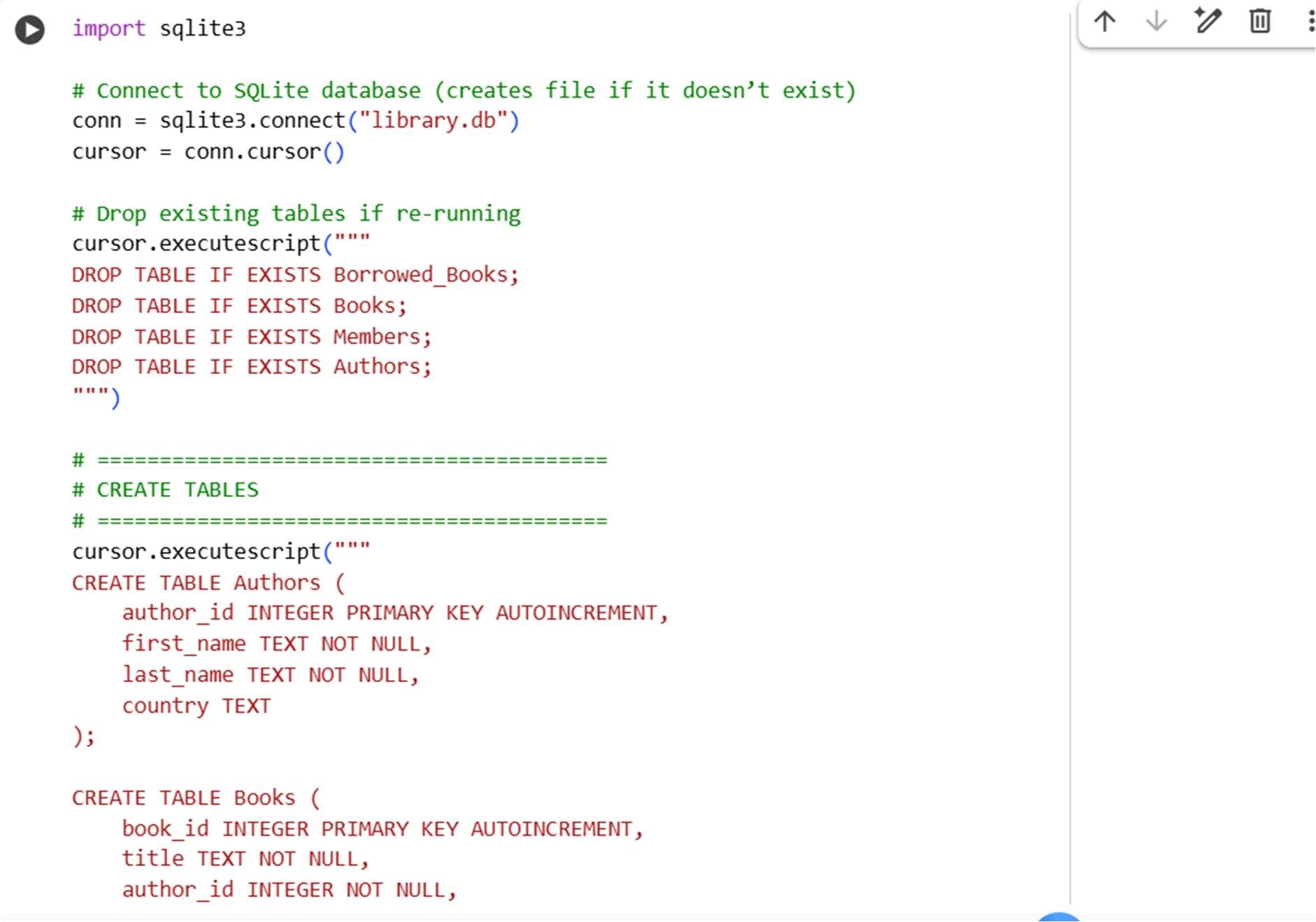
'Alice', 'alice@example.com'), (2, 'Bob',

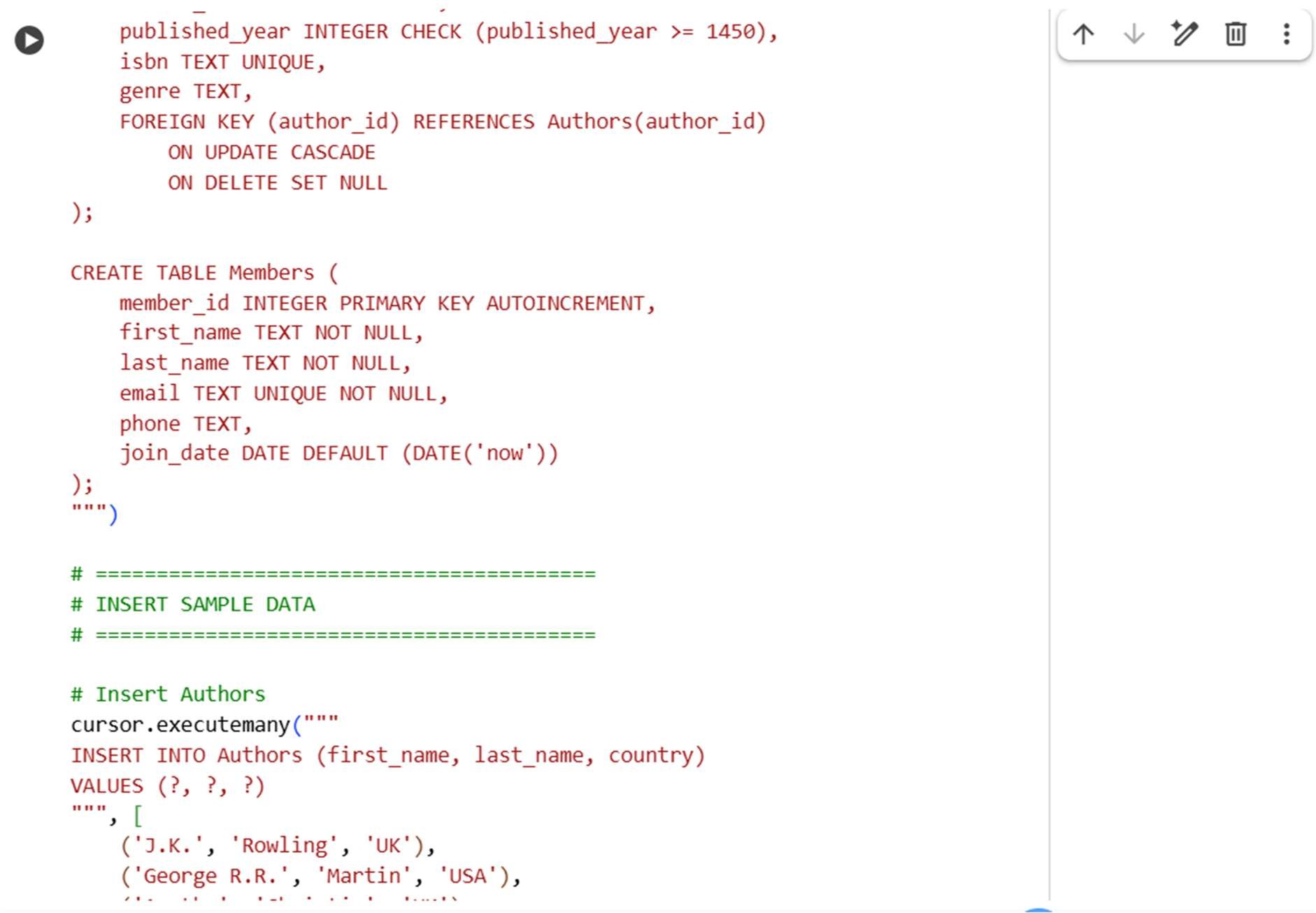
'bob@example.com'), (3, 'Charlie',

'charlie@example.com');

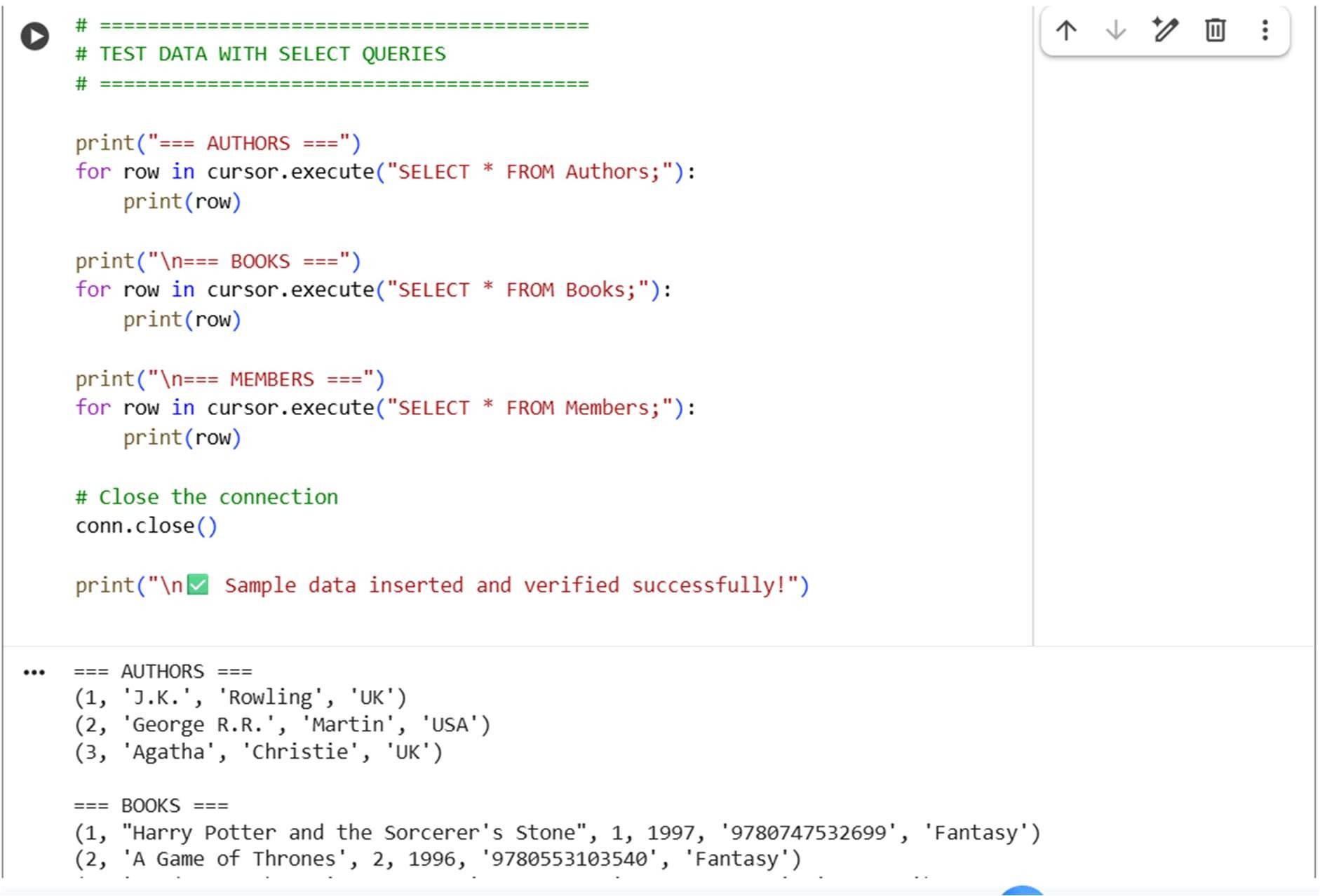
Expected Output:

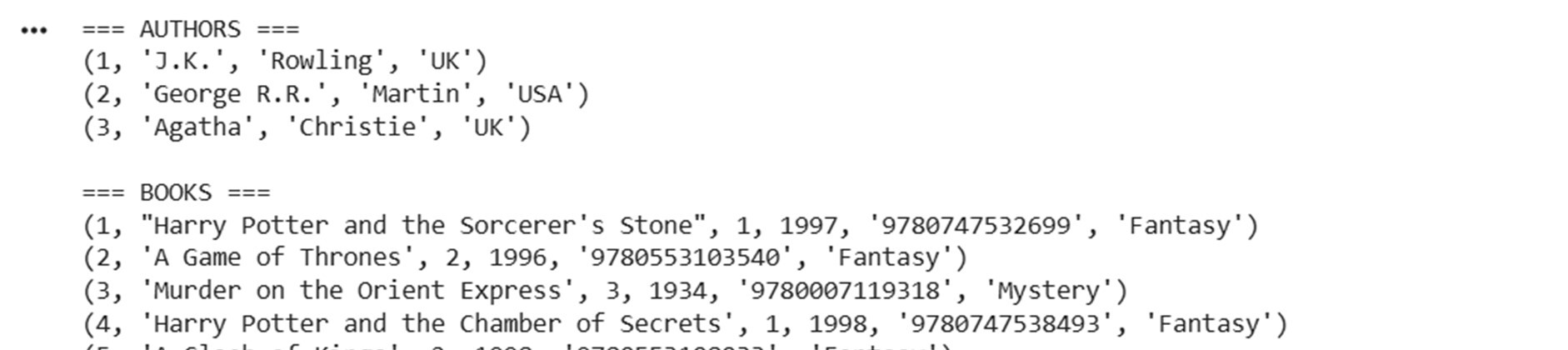
* Sample data inserted correctly.
* SELECT \* FROM Books; shows all 5 books.
* SELECT \* FROM Members; shows all 3 members



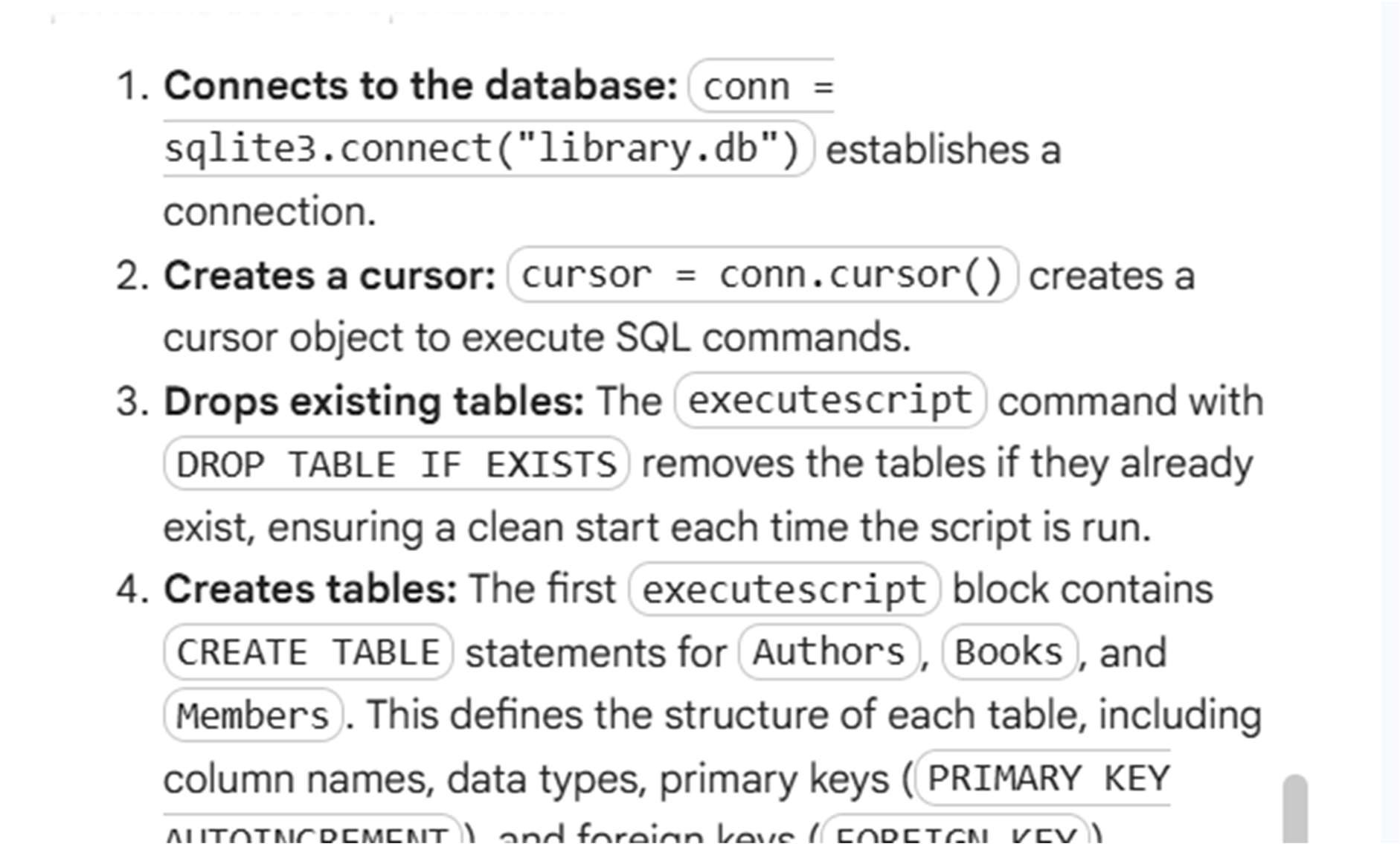


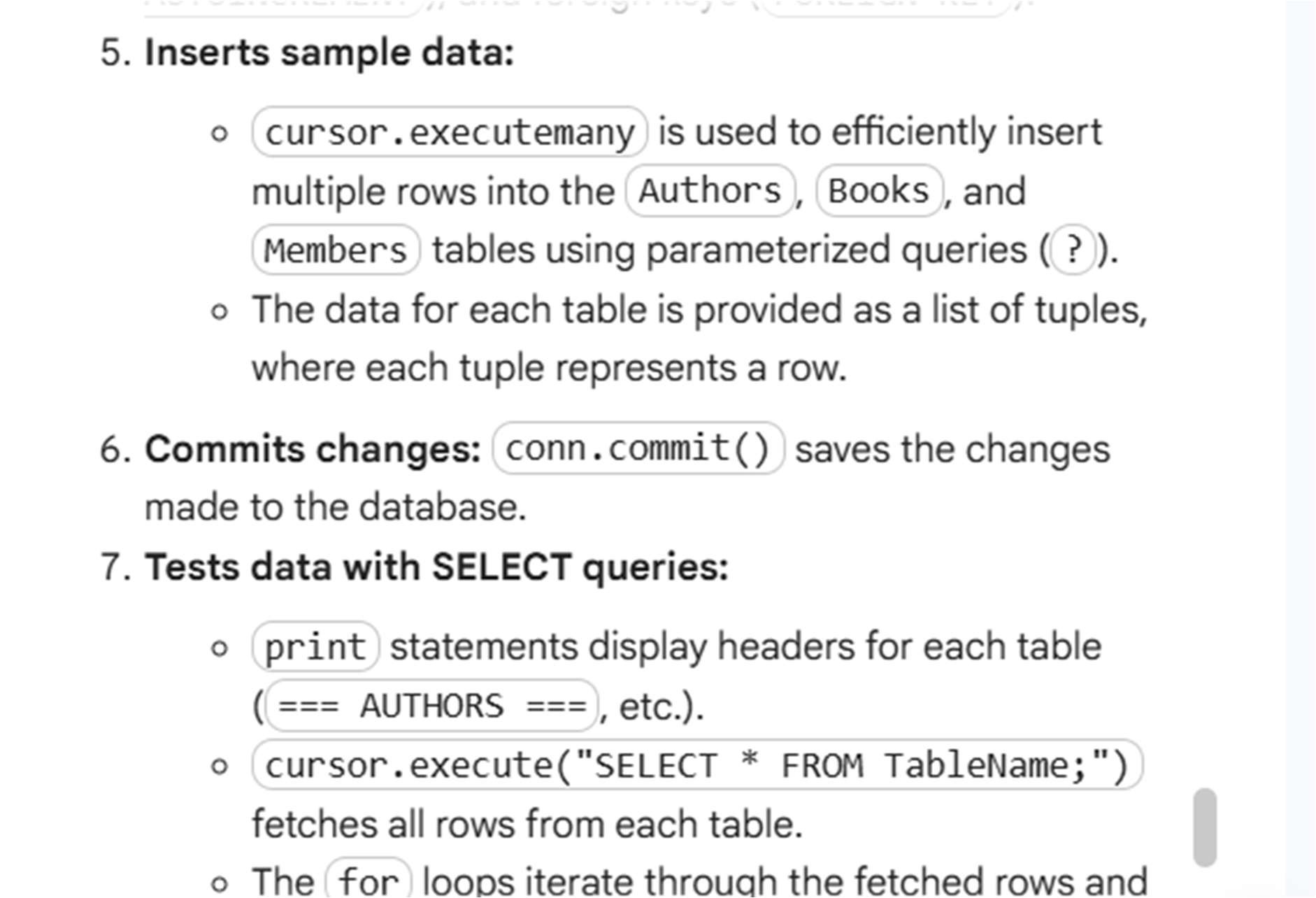


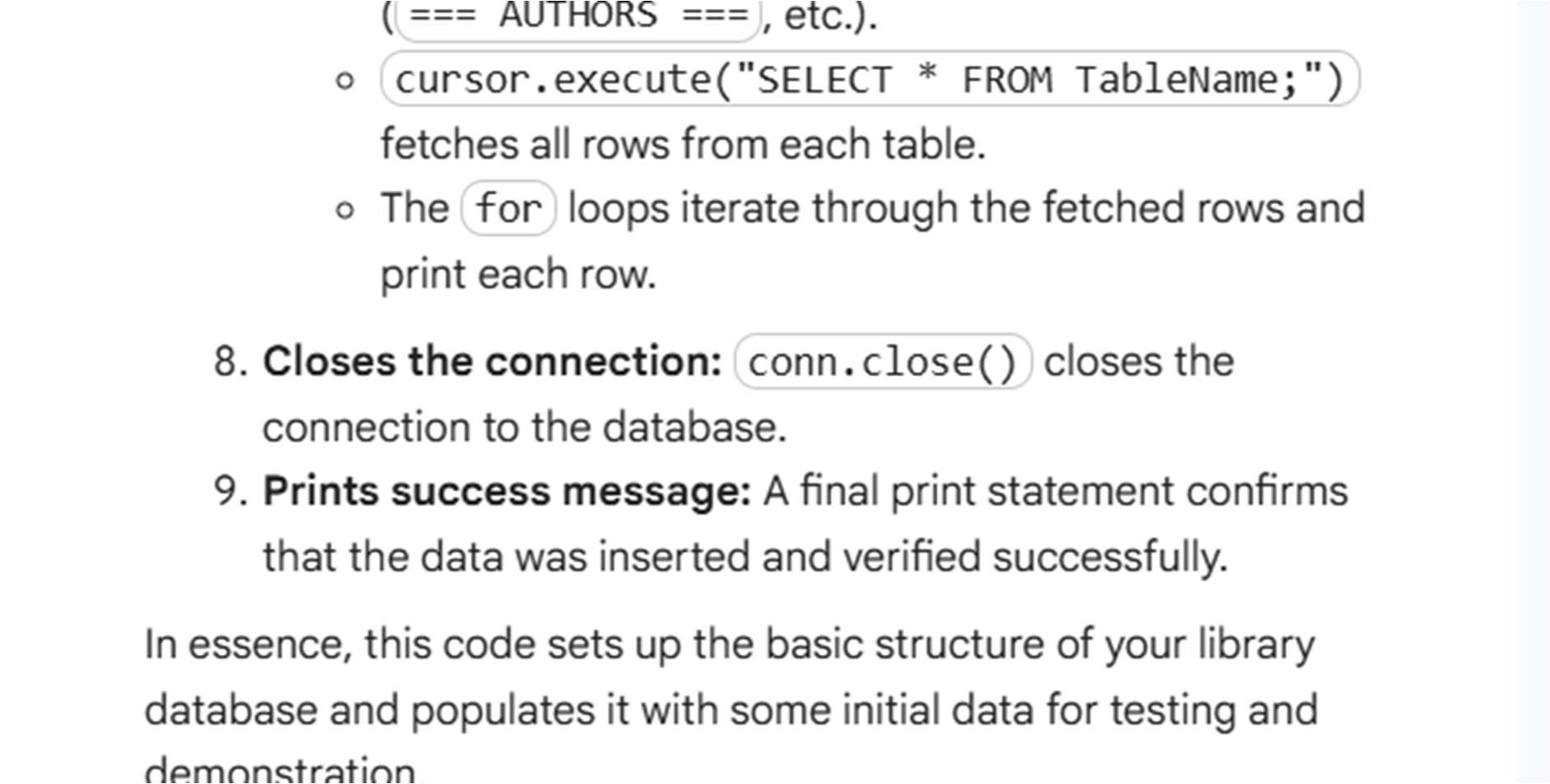




EXPLANATION:







Task 3: Basic CRUD Operations Instructions:

* Use AI to generate CRUD queries:
* Read all books.
* Update a member’s email.
* Delete a book.

• Verify correctness using SELECT statements.

Starter Code Example:

-- Read all books

SELECT \* FROM Books;

-- Update a member's email

UPDATE Members

SET email = 'alice\_new@example.com'

WHERE member\_id = 1;

-- Delete a book by ID

DELETE FROM Books WHERE book\_id = 5;

PROMPT:

develop a python code to generate Use AI to generate CRUD queries: o Read all books. o Update a member’s email. o Delete a book. • Verify correctness using SELECT statements. Starter

Code Example: -- Read all books SELECT \* FROM

Books; -- Update a member's email UPDATE

Members SET email = 'alice\_new@example.com'

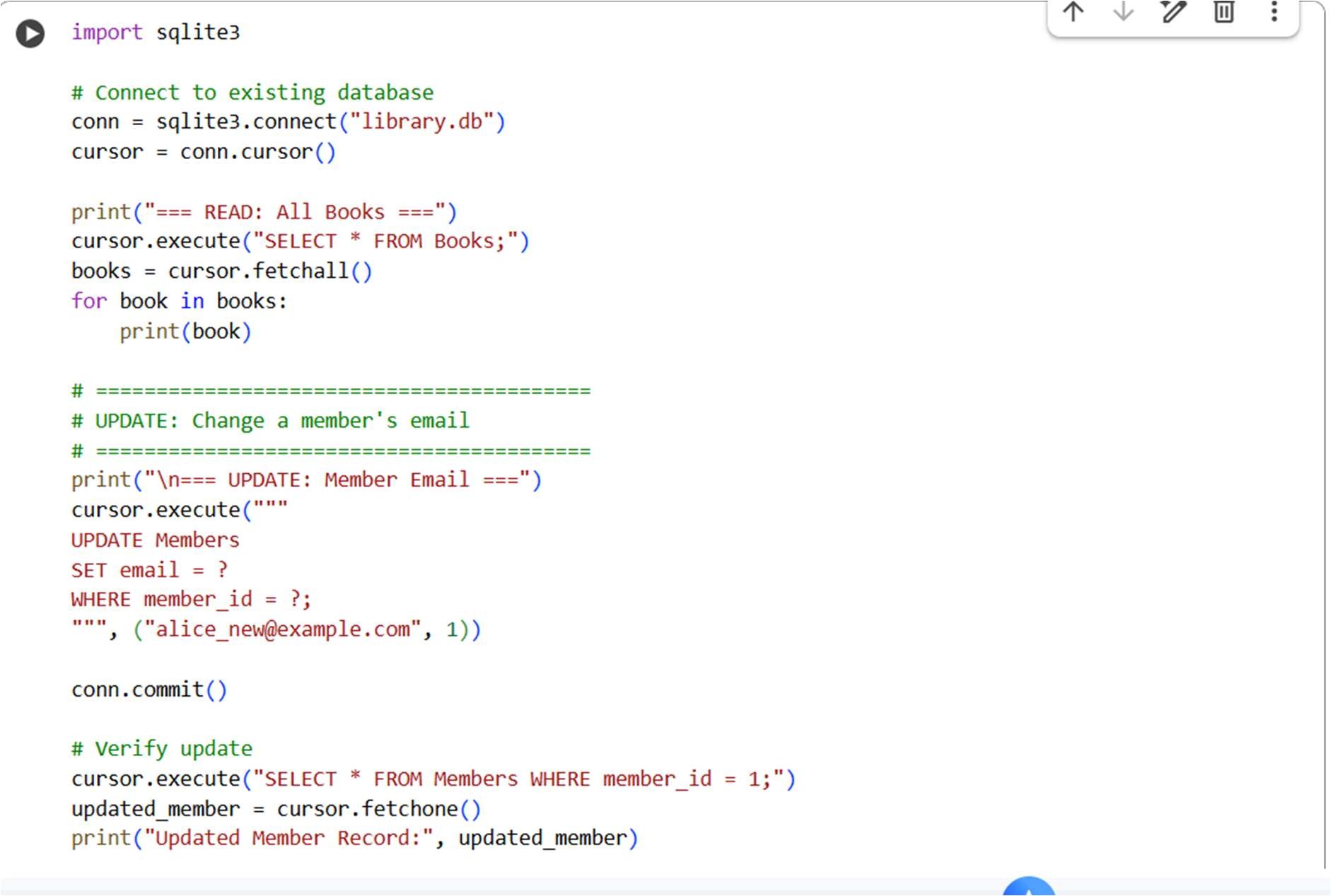
WHERE member\_id = 1; -- Delete a book by ID

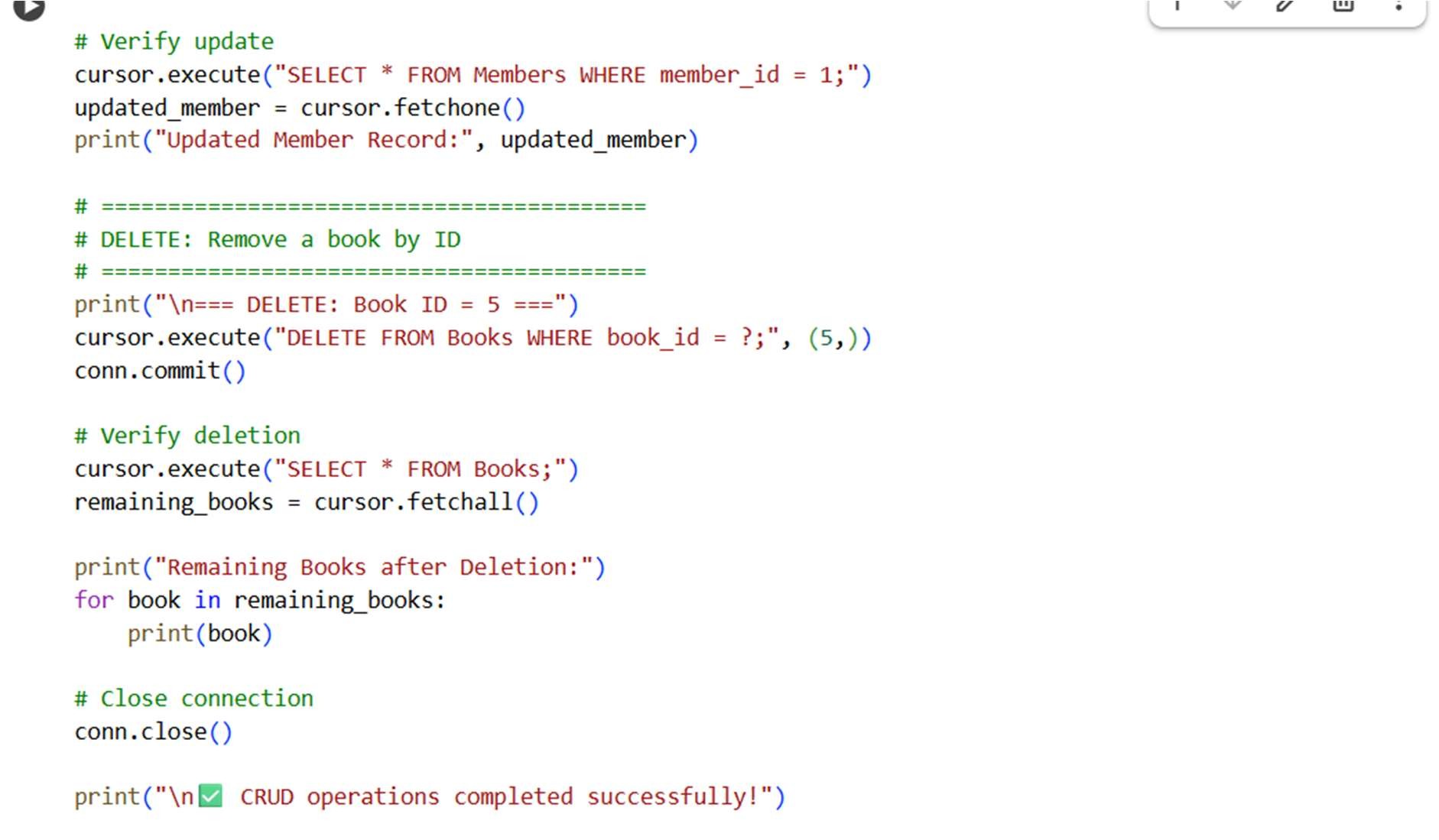
DELETE FROM Books WHERE book\_id = 5;

Expected Output: • Updated email is reflected in the Members table. • Deleted book is no longer in the Books table

Expected Output:

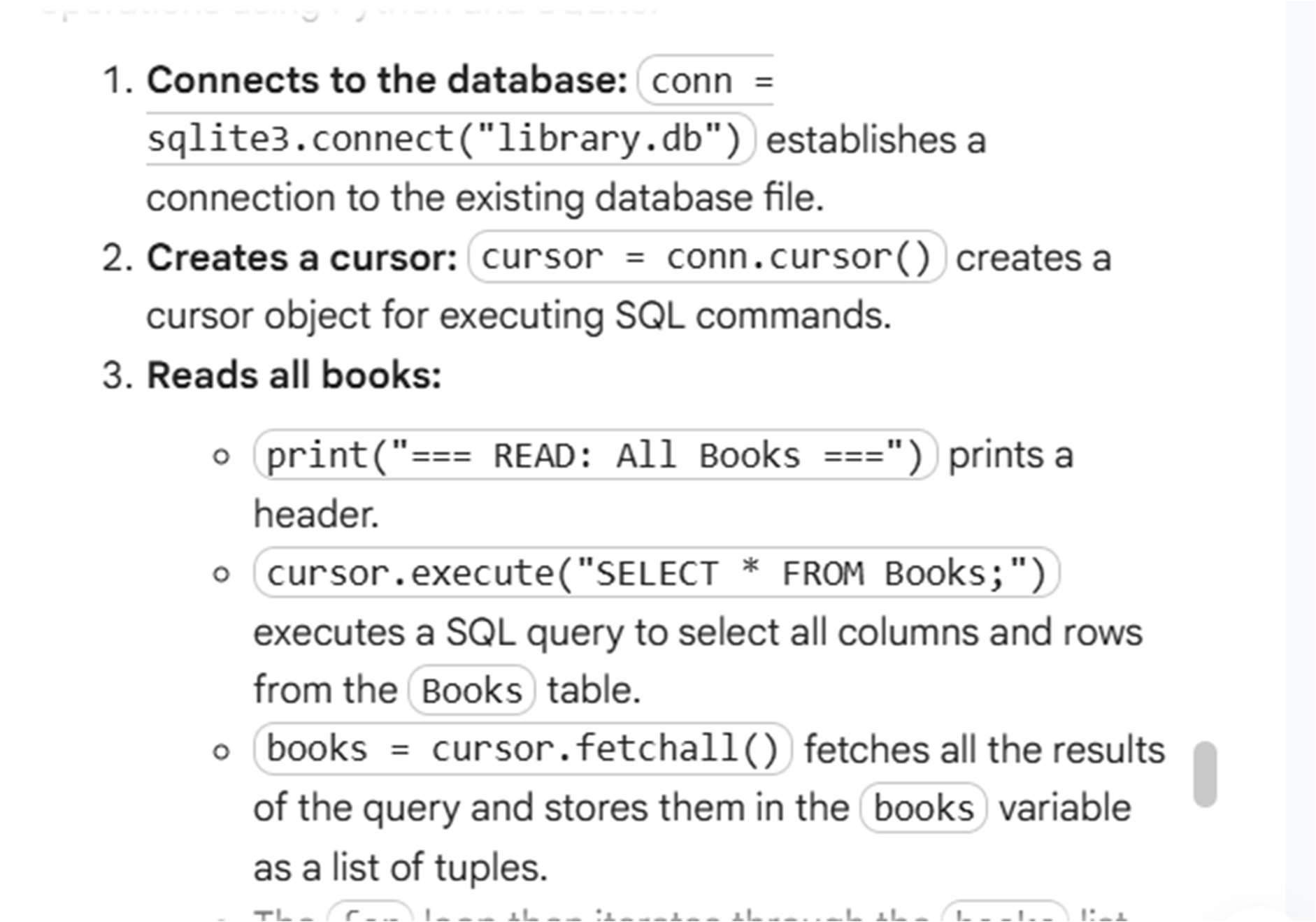
* Updated email is reflected in the Members table.
* Deleted book is no longer in the Books table

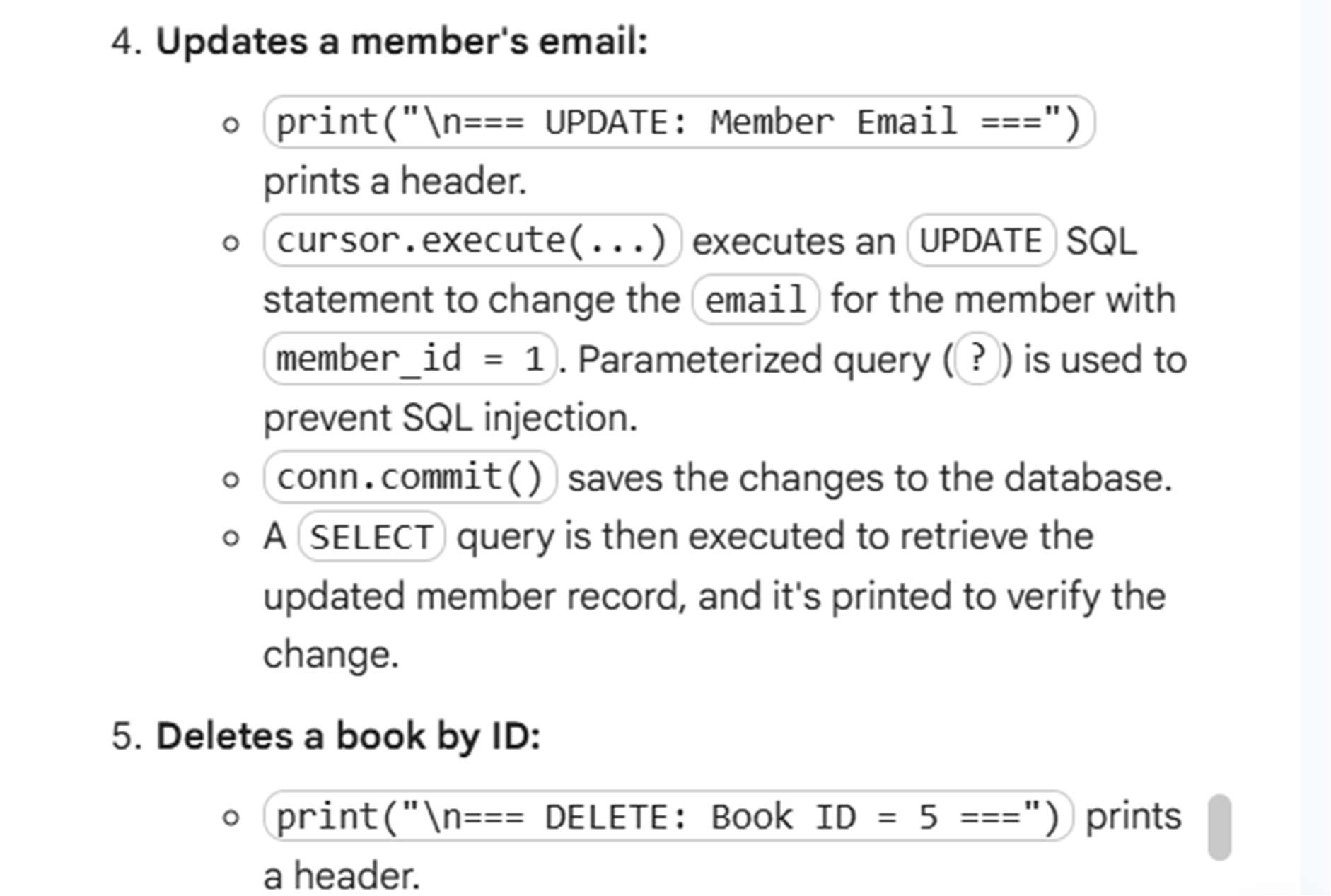


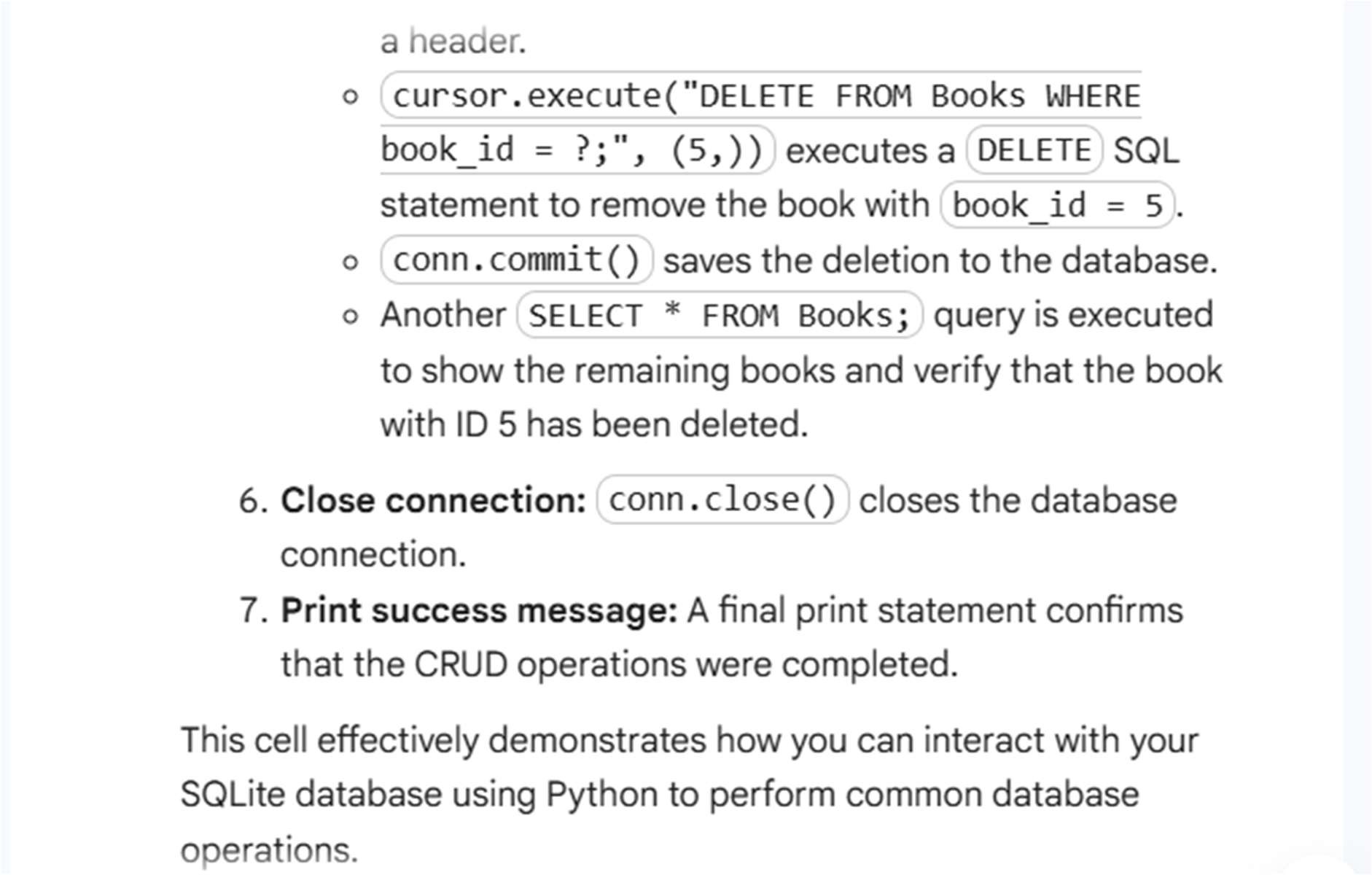




EXPLANATION:







Task 4: Aggregate Queries Instructions:

* Use AI tools to generate aggregate queries:
* Count books per author.
* List authors with more than one book.

• Analyze the results for correctness.

Starter Code Example:

-- Count books per author

SELECT a.name, COUNT(b.book\_id) AS book\_count

FROM Authors a

JOIN Books b ON a.author\_id = b.author\_id

GROUP BY a.name;

-- Authors with more than 1 book

SELECT a.name

FROM Authors a

JOIN Books b ON a.author\_id = b.author\_id

GROUP BY a.name HAVING COUNT(b.book\_id) > 1

PROMPT:

develop a code to generate Aggregate Queries Instructions: • Use AI tools to generate aggregate queries: o Count books per author. o List authors with more than one book. • Analyze the results for correctness. Starter Code Example: -- Count books per author SELECT a.name, COUNT(b.book\_id) AS book\_count FROM Authors a JOIN Books b ON

a.author\_id = b.author\_id GROUP BY a.name; -- Authors with more than 1 book SELECT a.name FROM Authors a JOIN Books b ON a.author\_id = b.author\_id GROUP BY a.name HAVING COUNT(b.book\_id) > 1

Expected Output:

Name Book\_Count

J.K. Rowling 2

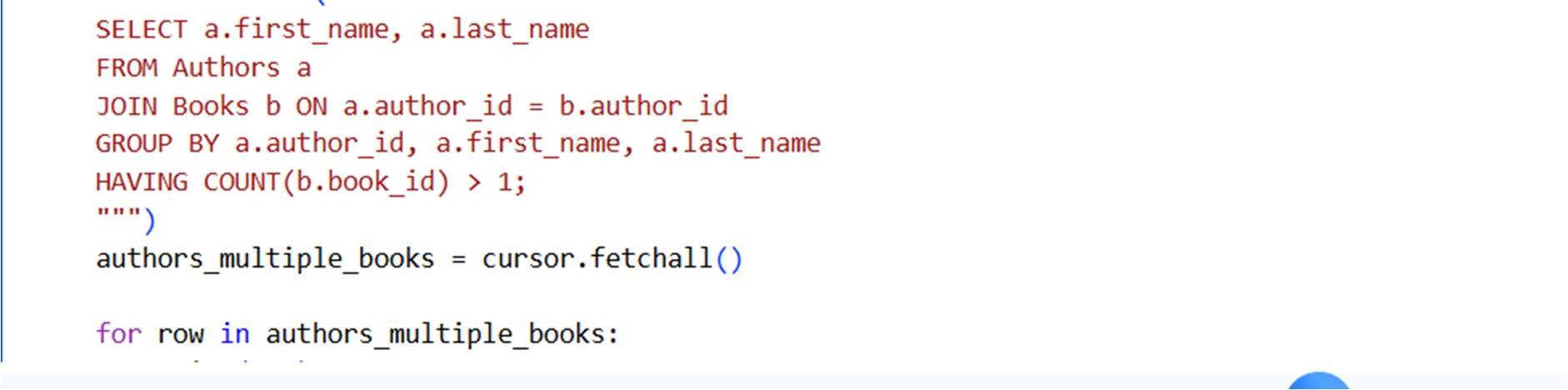
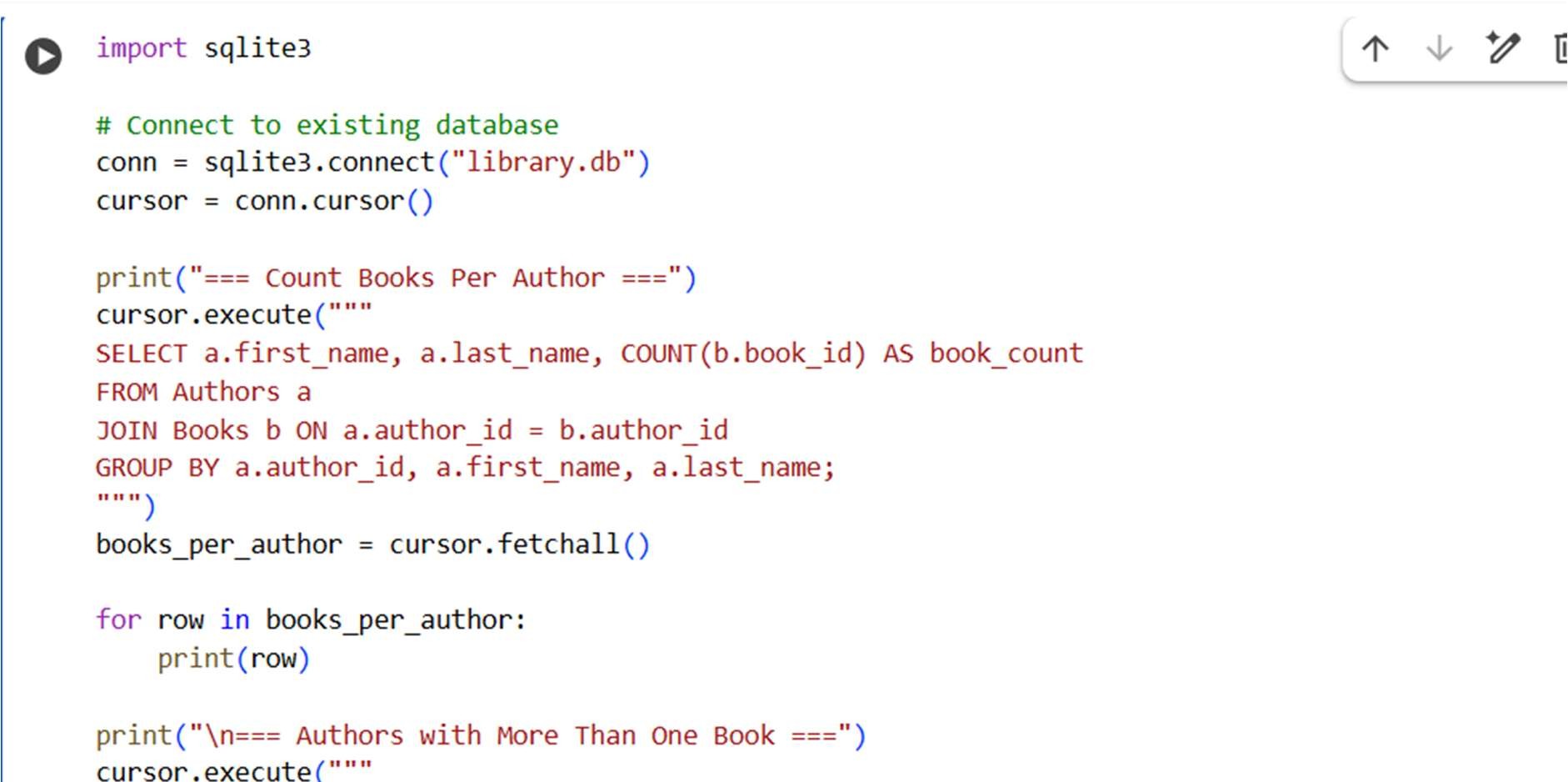
George R.R. Martin 2

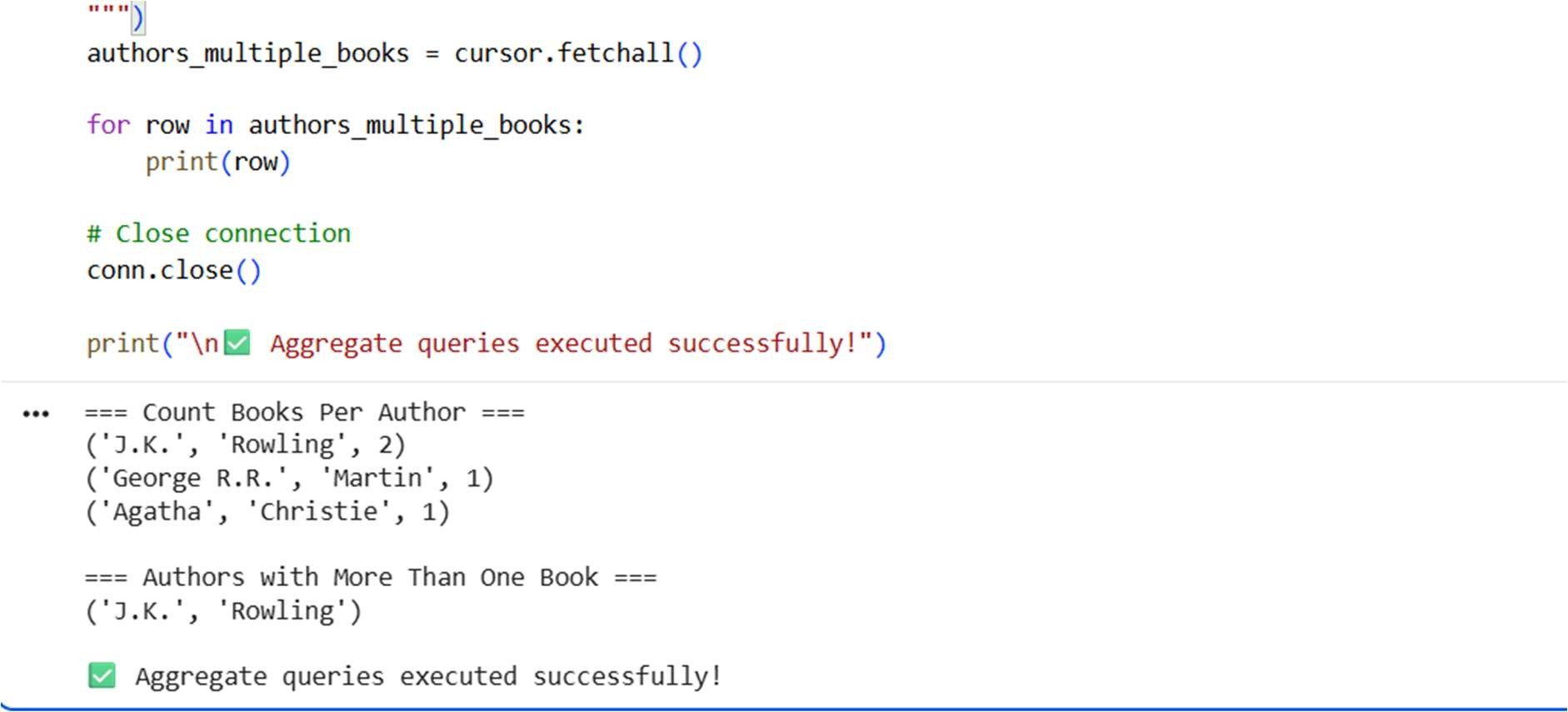
Agatha Christie 1

* Authors with more than 1 book:

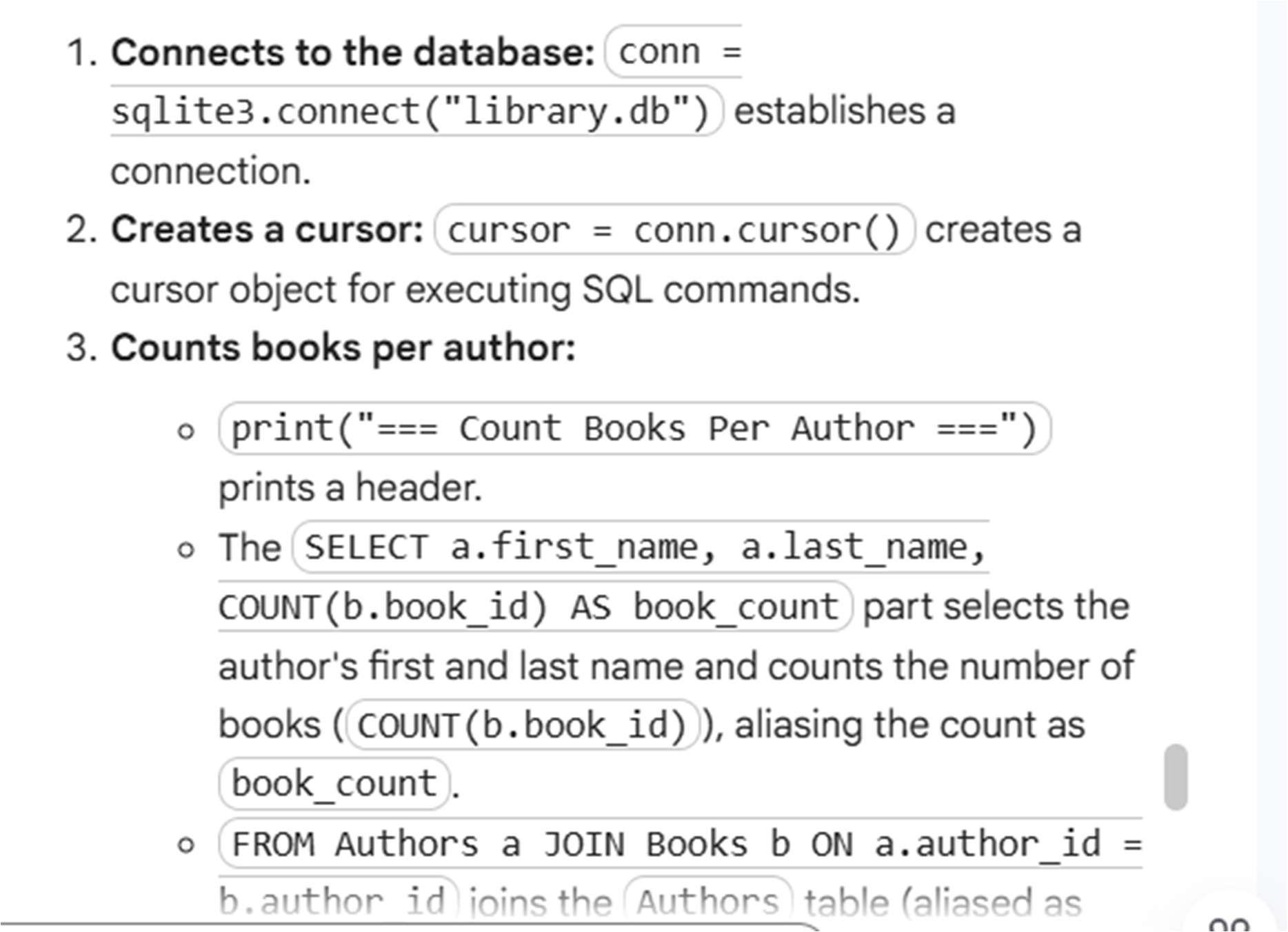
J.K. Rowling

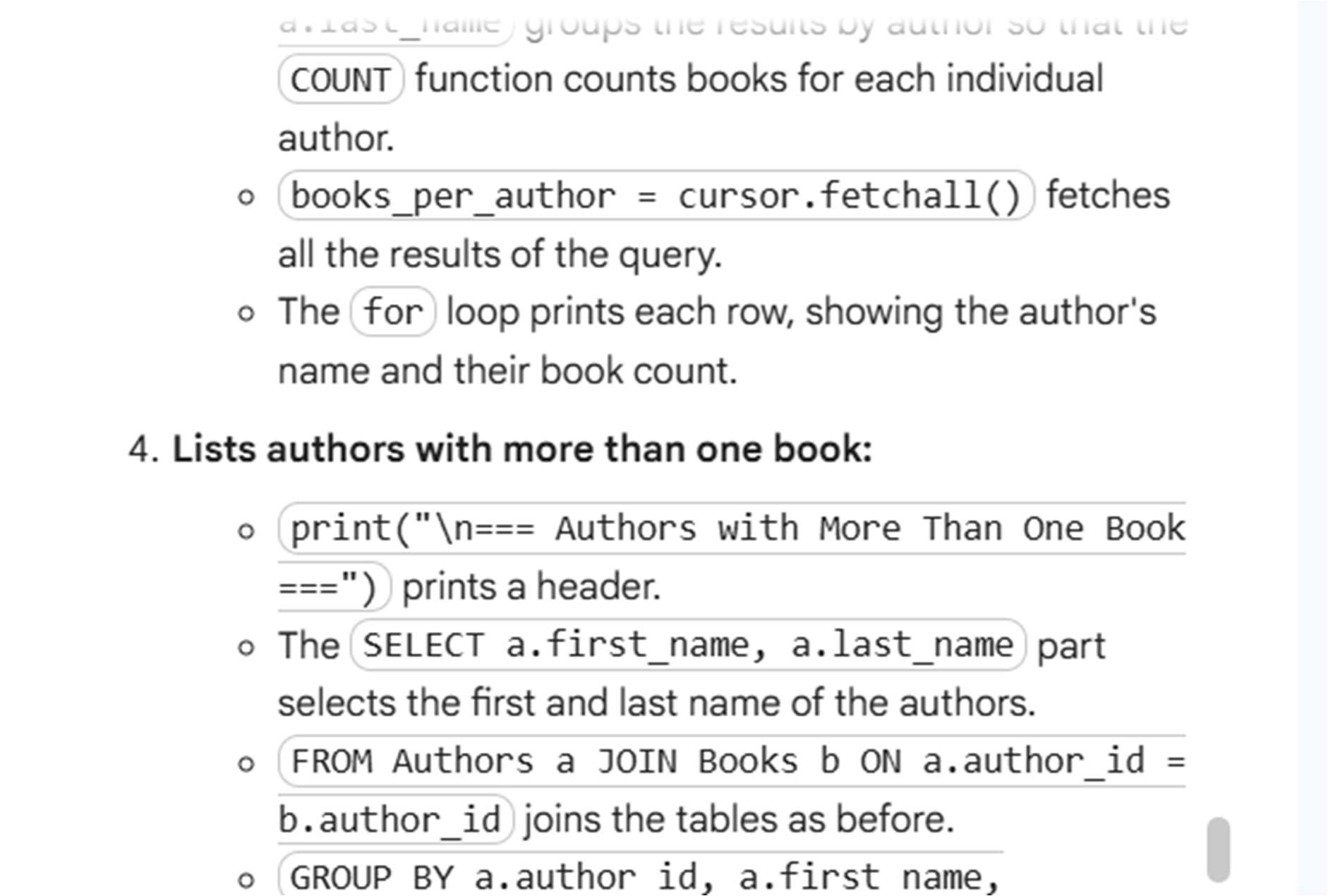
George R.R. Martin

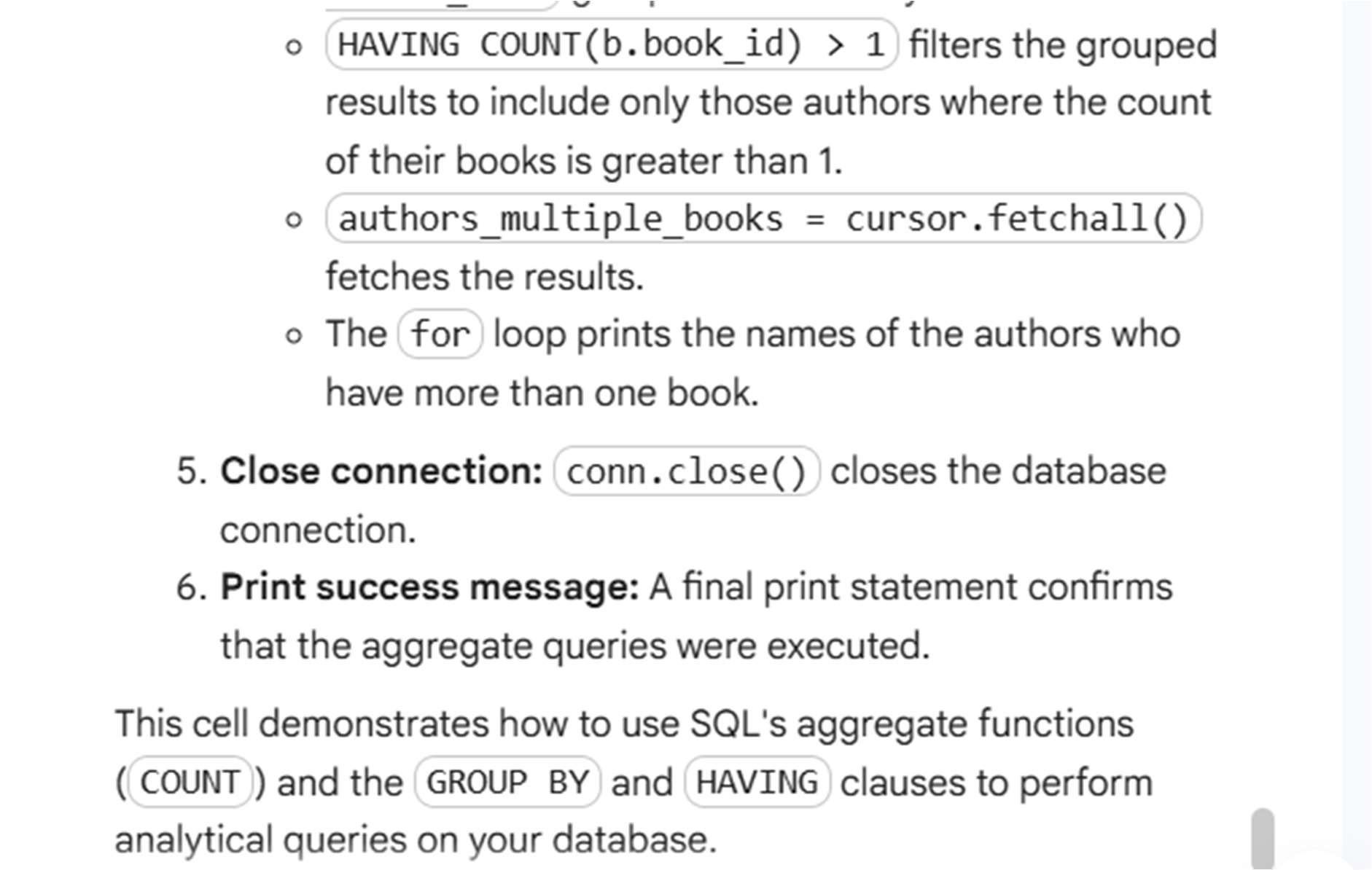




EXPLANATION:







Task 5: Optimize Queries Instructions:

* Use AI tools to analyze query efficiency.
* Optimize inefficient queries using joins, indexes, or reducing subqueries.
* Test optimized queries for correctness.

Starter Code Example:

-- Original query

SELECT \* FROM Books WHERE author\_id IN

(SELECT author\_id

FROM Authors WHERE country='UK');

-- Optimized query

SELECT b.\*

FROM Books b

JOIN Authors a ON b.author\_id = a.author\_id

WHERE a.country = 'UK';

PROMPT:

develop a code to generate python code Optimize Queries Instructions: • Use AI tools to analyze query efficiency. • Optimize inefficient queries using joins, indexes, or reducing subqueries. • Test optimized queries for correctness. Starter Code Example: -- Original query SELECT \* FROM Books

WHERE author\_id IN (SELECT author\_id FROM

Authors WHERE country='UK'); -- Optimized query

SELECT b.\* FROM Books b JOIN Authors a ON

b.author\_id = a.author\_id WHERE a.country = 'UK';

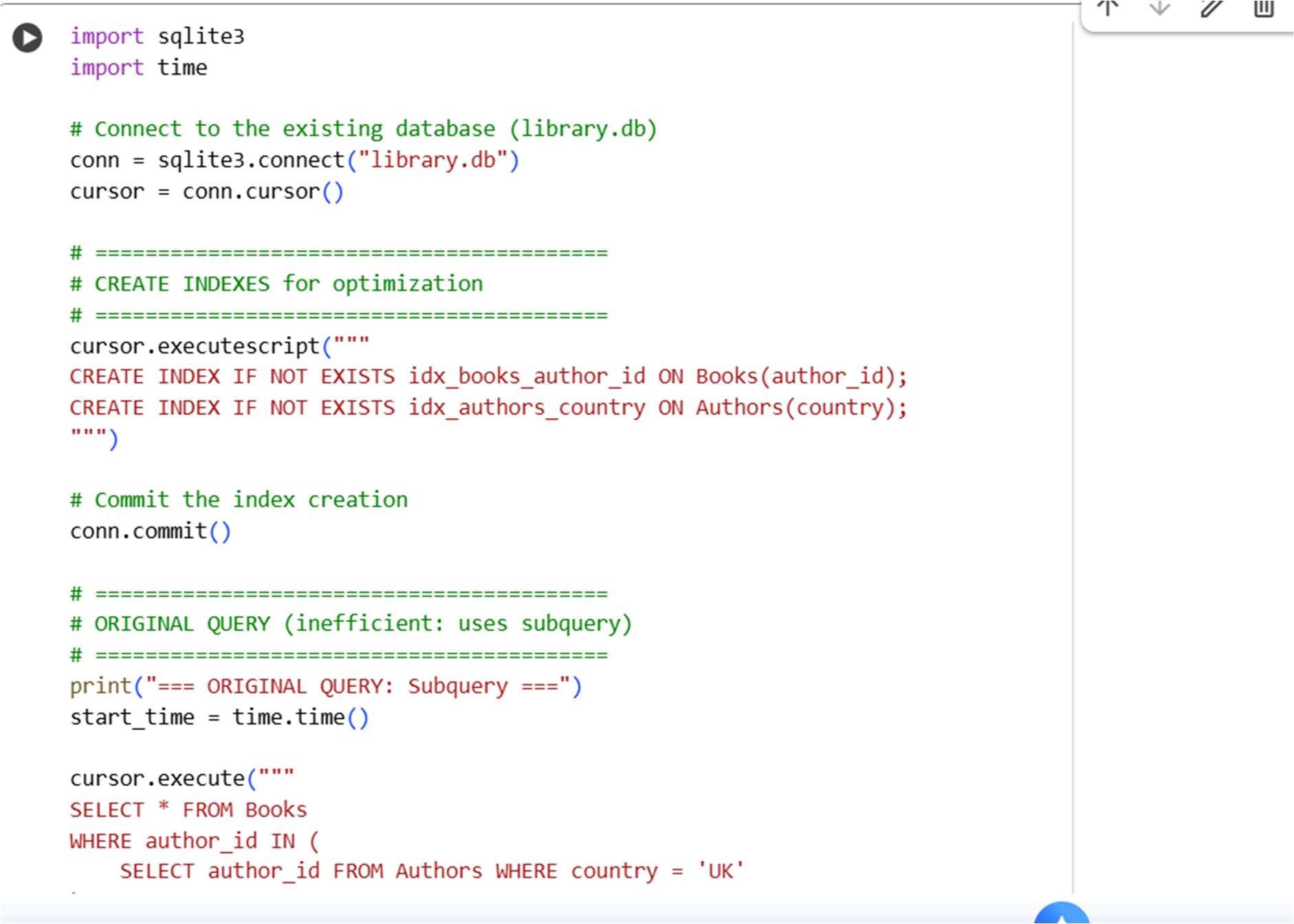
Expected Output:

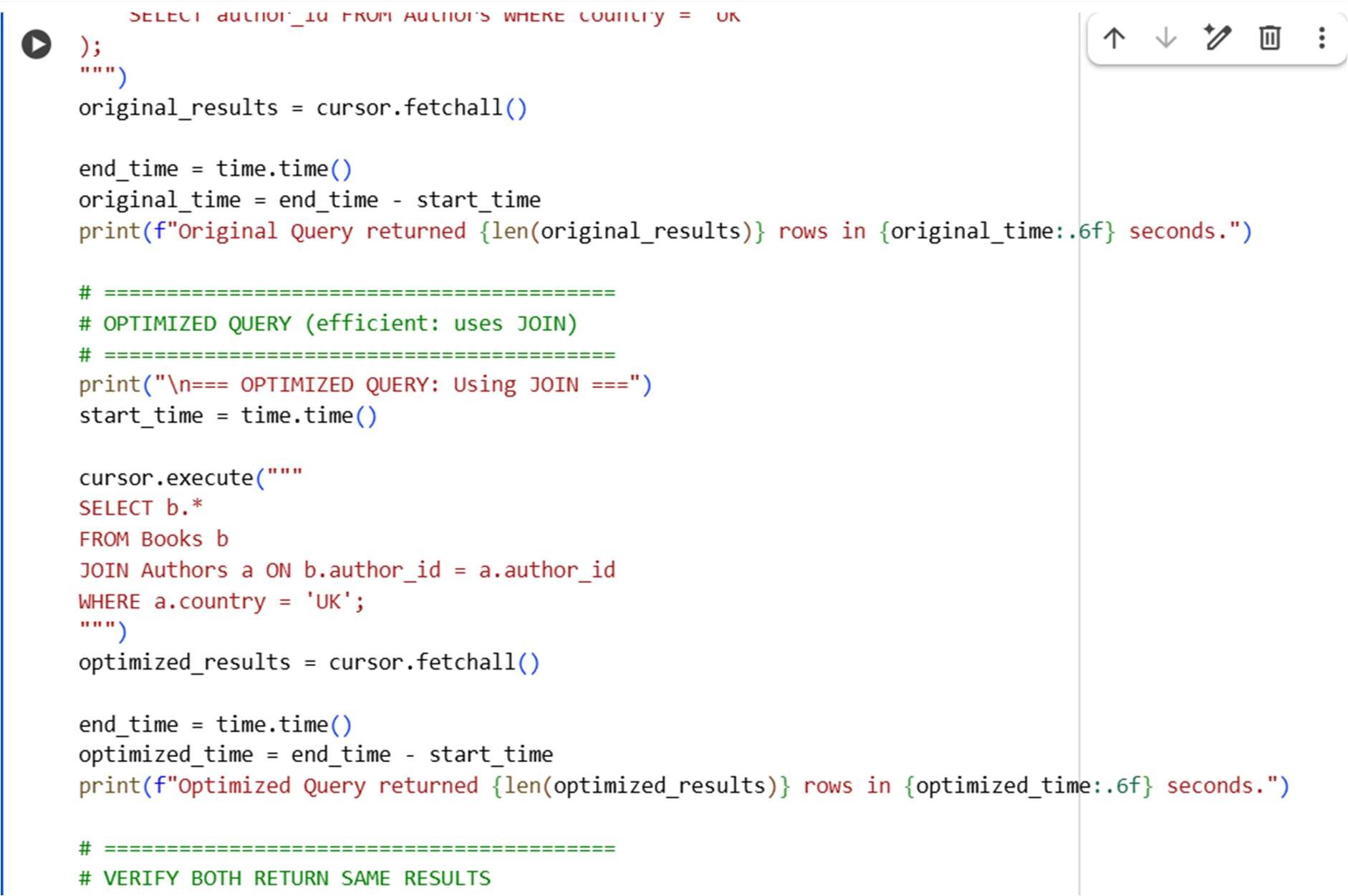
* Both queries return the same result: all books written by UK authors.
* Optimized query performs faster for larger datasets.

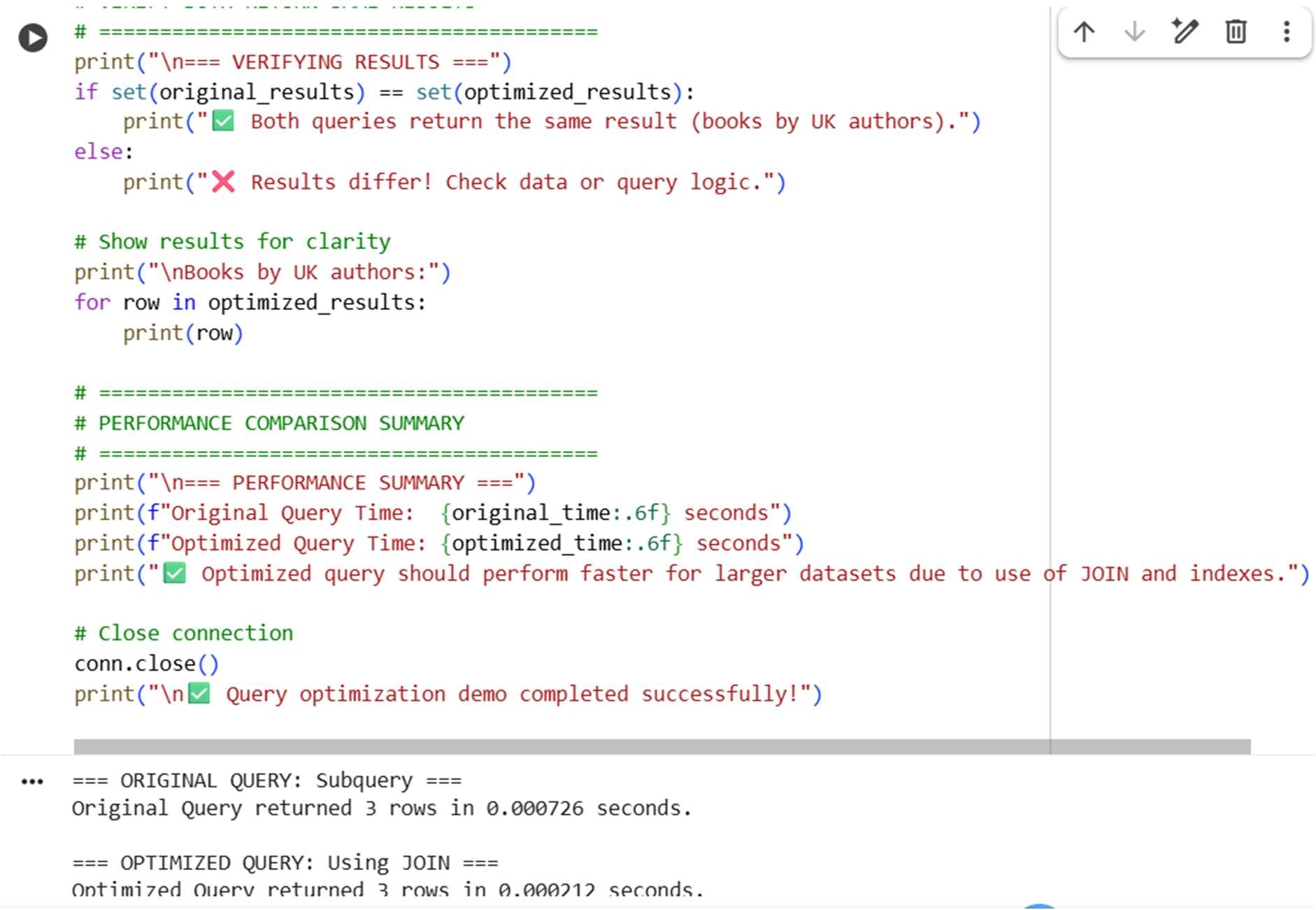
This lab allows students to use AI tools like

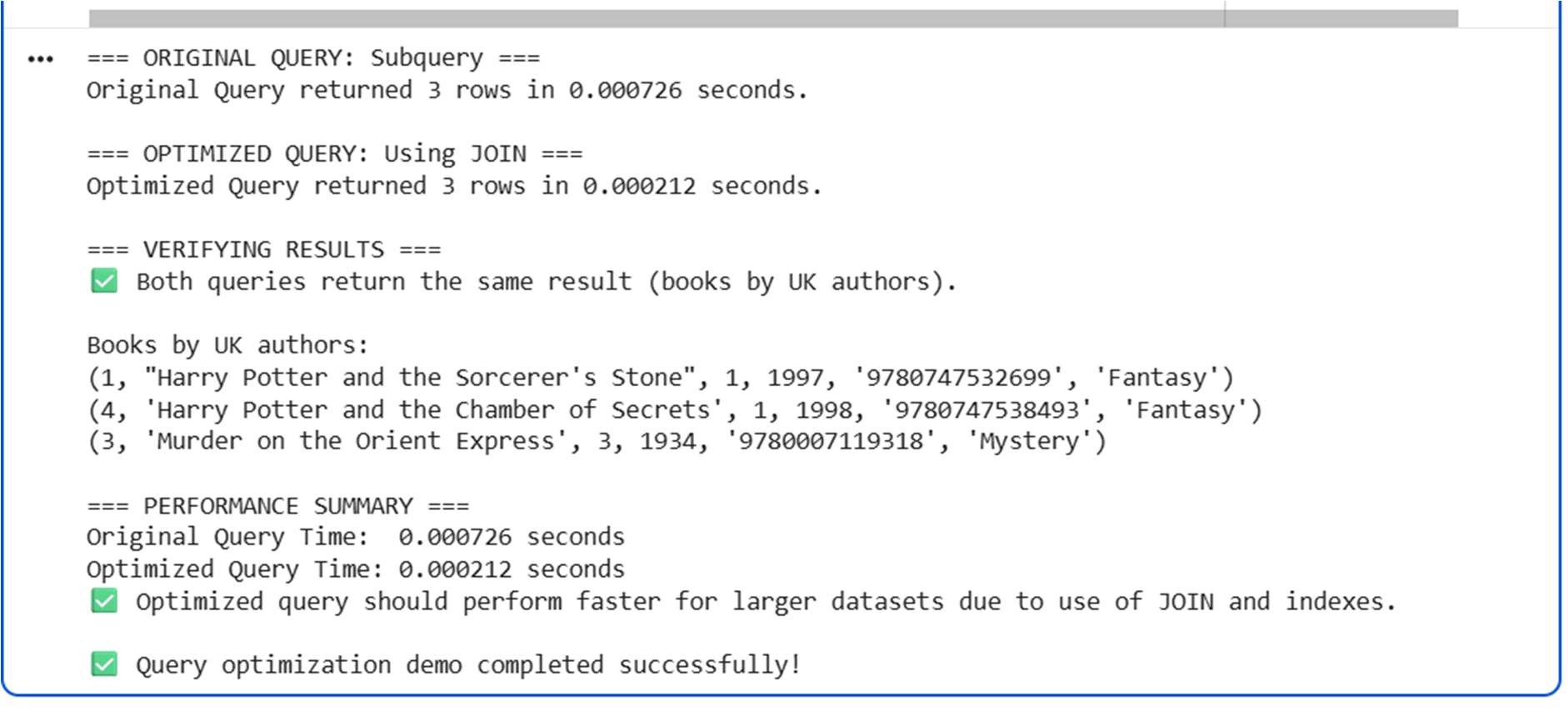
Copilot, Cursor, or Gemini to:

* Generate schema definitions
* Auto-complete insert statements
* Write CRUD and aggregate SQL queries
* Optimize queries and ensure proper normalization









EXPLANATION:

