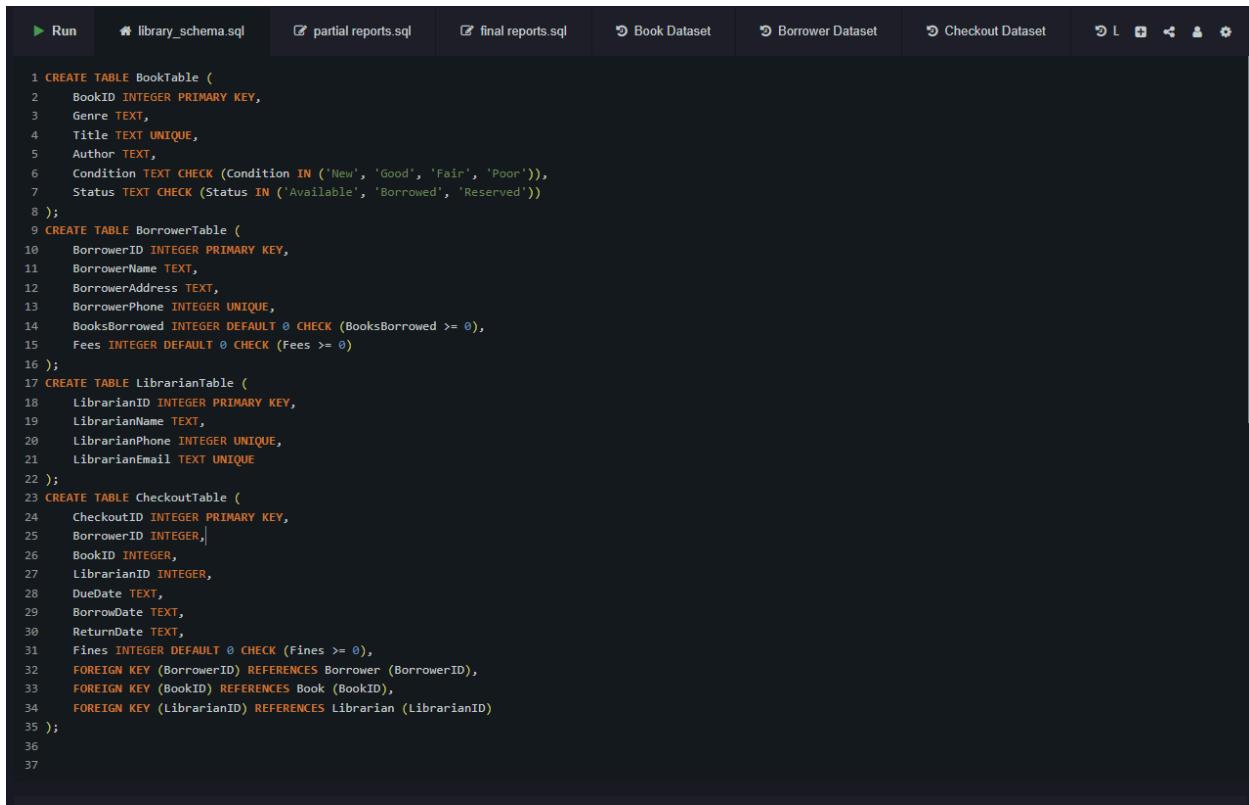


## Step 1

<https://dbdiagram.io/d/Project-2-ER-Diagram-Jeffery-Yong-68b9cdec61a46d388e81e634>

## Step 2

SS of code -



The screenshot shows a SQL editor interface with the following details:

- Toolbar buttons: Run, library\_schema.sql, partial reports.sql, final reports.sql, Book Dataset, Borrower Dataset, Checkout Dataset, and a settings gear icon.
- Code area content:

```
1 CREATE TABLE BookTable (
2     BookID INTEGER PRIMARY KEY,
3     Genre TEXT,
4     Title TEXT UNIQUE,
5     Author TEXT,
6     Condition TEXT CHECK (Condition IN ('New', 'Good', 'Fair', 'Poor')),
7     Status TEXT CHECK (Status IN ('Available', 'Borrowed', 'Reserved'))
8 );
9 CREATE TABLE BorrowerTable (
10    BorrowerID INTEGER PRIMARY KEY,
11    BorrowerName TEXT,
12    BorrowerAddress TEXT,
13    BorrowerPhone INTEGER UNIQUE,
14    BooksBorrowed INTEGER DEFAULT 0 CHECK (BooksBorrowed >= 0),
15    Fees INTEGER DEFAULT 0 CHECK (Fees >= 0)
16 );
17 CREATE TABLE LibrarianTable (
18    LibrarianID INTEGER PRIMARY KEY,
19    LibrarianName TEXT,
20    LibrarianPhone INTEGER UNIQUE,
21    LibrarianEmail TEXT UNIQUE
22 );
23 CREATE TABLE CheckoutTable (
24    CheckoutID INTEGER PRIMARY KEY,
25    BorrowerID INTEGER,
26    BookID INTEGER,
27    LibrarianID INTEGER,
28    DueDate TEXT,
29    BorrowDate TEXT,
30    ReturnDate TEXT,
31    Fines INTEGER DEFAULT 0 CHECK (Fines >= 0),
32    FOREIGN KEY (BorrowerID) REFERENCES Borrower (BorrowerID),
33    FOREIGN KEY (BookID) REFERENCES Book (BookID),
34    FOREIGN KEY (LibrarianID) REFERENCES Librarian (LibrarianID)
35 );
36
37
```

## Step 3

SS of import -

The screenshot shows a database management interface with several tabs at the top: Book Dataset, Borrower Dataset, Checkout Dataset, and Librarian Dataset. Below the tabs, there are code snippets for each dataset:

- Book Dataset:** Contains SQL code for creating a table named Book with columns BookID (TEXT), Genre (TEXT), Title (TEXT), Author (TEXT), Condition (TEXT), and Status (TEXT). It includes an INSERT statement with five rows of data.
- Borrower Dataset:** Contains SQL code for creating a table named Borrower with columns BorrowerID (TEXT), Name (TEXT), Address (TEXT), and Phone (TEXT).
- Checkout Dataset:** Contains SQL code for creating a table named Checkout with columns CheckoutID (TEXT), BorrowerID (TEXT), and Date (TEXT).
- Librarian Dataset:** Contains SQL code for creating a table named Librarian with columns LibrarianID (TEXT), Name (TEXT), and Address (TEXT).

On the right side, there are three additional code snippets:

- partial reports.sql:** Contains a DROP TABLE statement for the Book table.
- partial reports.sql:** Contains comments for basic library reports and a SELECT statement for the Book table.
- Book Dataset:** Contains the same SQL code as the Book Dataset tab.

Timestamps for the operations are shown on the right: 14:17:07, 14:17:03, 14:16:25, 14:16:19, 14:16:13, and 14:16:11.

## Imported Datasets

The screenshot shows a code editor with several tabs at the top: Run, library\_schema.sql, partial reports.sql, final reports.sql, Book Dataset, and another tab partially visible. The main area contains the following SQL code:

```
1 CREATE TABLE Book (BookID TEXT,Genre TEXT,Title TEXT,Author TEXT,Condition TEXT,Status TEXT);
2 INSERT INTO Book (BookID,Genre,Title,Author,Condition,Status) VALUES
3 ('1','Fantasy','Harry Potter and the Sorcerer's Stone','J.K. Rowling','New','Available'),
4 ('2','Dystopian','1984','George Orwell','Good','Borrowed'),
5 ('3','Romance','Pride and Prejudice','Jane Austen','Fair','Reserved'),
6 ('4','Classic','The Great Gatsby','F. Scott Fitzgerald','Good','Available'),
7 ('5','Mystery','The Hound of the Baskervilles','Arthur Conan Doyle','Good','Available');
```

Run library\_schema.sql partial reports.sql final reports.sql Book Dataset Borrower Dataset Checkout Dataset

```
1 CREATE TABLE Borrower (BorrowerID TEXT,BorrowerName TEXT,BorrowerAddress TEXT,BorrowerPhone TEXT,BooksBorrowed TEXT,Fees TEXT);
2 INSERT INTO Borrower (BorrowerID,BorrowerName,BorrowerAddress,BorrowerPhone,BooksBorrowed,Fees) VALUES
3 ('1','Andrew Nguyen','5 Oak Grove Ave','2221111','1','0'),
4 ('2','Kevin Li','16 Sutton Farm Rd','2227777','0','5'),
5 ('3','Mohit Sanagavarapu','34 Cedar Hill St','2223333','2','0'),
6 ('4','Nick Campanella','84 Birch Blvd','2229999','0','0');
```

Run -sql final reports.sql Book Dataset Borrower Dataset Checkout Dataset Librarian Datset views.sql

```
1 CREATE TABLE Checkout (CheckoutID TEXT,BorrowerID TEXT,BookID TEXT,LibrarianID TEXT,DueDate TEXT,BorrowDate TEXT,ReturnDate TEXT,Fines TEXT);
2 INSERT INTO Checkout (CheckoutID,BorrowerID,BookID,LibrarianID,DueDate,BorrowDate,ReturnDate,Fines) VALUES
3 ('1','1','2','1','2025-11-05','2025-10-22','2025-10-29','0'),
4 ('2','3','3','2','2025-10-20','2025-10-06','2025-10-25','2'),
5 ('3','2','4','3','2025-11-10','2025-10-26','NULL','0');
```

Run -sql final reports.sql Book Dataset Borrower Dataset Checkout Dataset Librarian Datset views.sql

```
1 CREATE TABLE Librarian (LibrarianID TEXT,LibrarianName TEXT,LibrarianPhone TEXT,LibrarianEmail TEXT);
2 INSERT INTO Librarian (LibrarianID,LibrarianName,LibrarianPhone,LibrarianEmail) VALUES
3 ('1','Jeffery Yong','1234567','jyong@library.com'),
4 ('2','Shiv Manhas','2345678','smanhas@library.com'),
5 ('3','Jack Hennelly','3456789','jhennelly@library.com');
```

## Step 4 Code -

▶ Run schema.sql    ⚙ partial reports.sql    ⚙ final reports.sql    ⏪ Book Dataset

```
1 -- 1. Show all books in the library
2 SELECT * FROM Book;
3
4 -- 2. Show all borrowers in the system
5 SELECT * FROM Borrower;
6
7 -- 3. Show all librarians who work at the library
8 SELECT * FROM Librarian;
9
10 -- 4. Show all checkouts that have been recorded
11 SELECT * FROM Checkout;
12
13 -- 5. Show only books that are currently available
14 SELECT Title, Author, Status
15 FROM Book
16 WHERE Status = 'Available';
17
18 -- 6. Show the names and phone numbers of borrowers
19 SELECT BorrowerName, BorrowerPhone
20 FROM Borrower;
21
22 -- 7. Show books in a specific genre
23 SELECT Title, Genre
24 FROM Book
25 WHERE Genre = 'Classic';
26
27 -- 8. Show each borrower's name and how many books they've borrowed
28 SELECT Borrower.BorrowerName, COUNT(Checkout.CheckoutID) AS TotalBorrowed
29 FROM Borrower
30 LEFT JOIN Checkout ON Borrower.BorrowerID = Checkout.BorrowerID
31 GROUP BY Borrower.BorrowerName;
32
```

BookID	Genre	Title	Author	Condition	Status
1	Fantasy	Harry Potter and the Sorcerer's Stone	J.K. Rowling	New	Available
2	Dystopian	1984	George Orwell	Good	Borrowed
3	Romance	Pride and Prejudice	Jane Austen	Fair	Reserved
4	Classic	The Great Gatsby	F. Scott Fitzgerald	Good	Available
5	Mystery	The Hound of the Baskervilles	Arthur Conan Doyle	Good	Available

2-

BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees
1	Andrew Nguyen	5 Oak Grove Ave	2221111	1	0
2	Kevin Li	16 Sutton Farm Rd	2227777	0	5
3	Mohit Sanagavarapu	34 Cedar Hill St	2223333	2	0
4	Nick Campanella	84 Birch Blvd	2229999	0	0

3-

LibrarianID	LibrarianName	LibrarianPhone	LibrarianEmail
1	Jeffery Yong	1234567	jyong@library.com
2	Shiv Manhas	2345678	smanhas@library.com
3	Jack Hennelly	3456789	jhennelly@library.com

4-

CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines
1	1	2	1	2025-11-05	2025-10-22	2025-10-29	0
2	3	3	2	2025-10-20	2025-10-06	2025-10-25	2
3	2	4	3	2025-11-10	2025-10-26	NULL	0

5-

Title	Author	Status
Harry Potter and the Sorcerer's Stone	J.K. Rowling	Available
The Great Gatsby	F. Scott Fitzgerald	Available
The Hound of the Baskervilles	Arthur Conan Doyle	Available

6-

BorrowerName	BorrowerPhone
Andrew Nguyen	2221111
Assignment2LibrarianDatasetSheet1	2227777
Mohit Sanagavarapu	2223333
Nick Campanella	2229999

7-

Title	Genre
The Great Gatsby	Classic

8-

BorrowerName	TotalBorrowed
Andrew Nguyen	1
Kevin Li	1
Mohit Sanagavarapu	1
Nick Campanella	0

Step 5

Code -

▶ Run    library\_schema.sql    partial reports.sql    final reports.sql    Book Data

```
1 -- 9. Show the names of borrowers who still have books checked out
2 SELECT DISTINCT Borrower.BorrowerName
3 FROM Checkout
4 JOIN Borrower ON Checkout.BorrowerID = Borrower.BorrowerID
5 WHERE Checkout.ReturnDate IS NULL;
6
7 -- 10. Show how many checkouts each librarian has handled
8 SELECT
9   Librarian.LibrarianName,
10  COUNT(Checkout.CheckoutID) AS TotalCheckouts
11 FROM Checkout
12 JOIN Librarian ON Checkout.LibrarianID = Librarian.LibrarianID
13 GROUP BY Librarian.LibrarianName;
14
15 -- 11. Show which book has been borrowed the most
16 SELECT
17   Book.Title,
18  COUNT(Checkout.BookID) AS TimesBorrowed
19 FROM Checkout
20 JOIN Book ON Checkout.BookID = Book.BookID
21 GROUP BY Book.Title
22 ORDER BY TimesBorrowed DESC
23 LIMIT 1;
24
25 -- 12. Show all borrowers who owe fees
26 SELECT BorrowerName, Fees
27 FROM Borrower
28 WHERE Fees > 0;
29
30 -- 13. Show the title, condition, and status of every book
31 SELECT Title, Condition, Status
32 FROM Book;
33
34 -- 14. Show how many books there are in each genre
35 SELECT Genre, COUNT(BookID) AS TotalBooks
36 FROM Book
37 GROUP BY Genre;
38
39 -- 15. Show the average fine for books that have been returned
40 SELECT ROUND(AVG(Fines), 2) AS AverageFine
41 FROM Checkout
42 WHERE ReturnDate IS NOT NULL;
43
```

9-

BorrowerName
Kevin Li

10-

LibrarianName	TotalCheckouts
Jack Hennelly	1
Jeffery Yong	1
Shiv Manhas	1

11-

Title	TimesBorrowed
The Great Gatsby	1

12-

BorrowerName	Fees
Kevin Li	5

13-

Title	Condition	Status
Harry Potter and the Sorcerer's ...	New	Available
1984	Good	Borrowed
Pride and Prejudice	Fair	Reserved
The Great Gatsby	Good	Available
The Hound of the Baskervilles	Good	Available

14-

Genre	TotalBooks
Classic	1
Dystopian	1
Fantasy	1
Mystery	1
Romance	1

15-

AverageFine
1

Views.sql

Code -

```

▶ Run   library_schema.sql   partial reports.sql   final reports.sql   Book Dataset   Borrower Dataset   Checkout Dataset   Librarian Dataset   views.sql

1 -- View 1 - v_patron_activity
2 CREATE VIEW v_patron_activity AS
3 SELECT
4   b.BorrowerID,
5   b.BorrowerName,
6   COUNT(c.CheckoutID) AS TotalBorrowed,
7   SUM(CASE WHEN c.ReturnDate IS NOT NULL THEN 1 ELSE 0 END) AS BooksReturned,
8   b.Fees AS CurrentFees
9 FROM Borrower b
10 LEFT JOIN Checkout c ON b.BorrowerID = c.BorrowerID
11 GROUP BY b.BorrowerID, b.BorrowerName;
12
13 -- view 2 - v_branch_performance
14
15 -- Create a view to measure librarian or branch performance
16 CREATE VIEW v_branch_performance AS
17 SELECT
18   l.LibrarianID,
19   l.LibrarianName,
20   COUNT(c.CheckoutID) AS TotalCheckouts,
21   SUM(c.Fines) AS TotalFinesCollected
22 FROM Librarian l
23 LEFT JOIN Checkout c ON l.LibrarianID = c.LibrarianID
24 GROUP BY l.LibrarianID, l.LibrarianName;
25
26 --view 3 - v_catalog_status
27
28 -- Create a view to show the current status of the book catalog
29 CREATE VIEW v_catalog_status AS
30 SELECT
31   BookID,
32   Title,
33   Author,
34   Genre,
35   Condition,
36   Status
37 FROM Book;
38
39 -- running the views
40
41 SELECT * FROM v_patron_activity;
42 SELECT * FROM v_branch_performance;
43 SELECT * FROM v_catalog_status;
44

```

### v\_patron\_activity

BorrowerID	BorrowerName	TotalBorrowed	BooksReturned	CurrentFees
1	Andrew Nguyen	1	1	0
2	Kevin Li	1	0	5
3	Mohit Sanagavar...	1	1	0
4	Nick Campanella	0	0	0

### v\_branch\_performance

LibrarianID	LibrarianName	TotalCheckouts	TotalFinesCollected
1	Jeffery Yong	1	0
2	Shiv Manhas	1	2
3	Jack Hennelly	1	0

### v\_catalog\_status

#	BookID	Title	Author	Genre	Condition	Status
1		Harry Potter ...	J.K. Rowling	Fantasy	New	Available
2		1984	George Orwell	Dystopian	Good	Borrowed
3		Pride and Pr...	Jane Austen	Romance	Fair	Reserved
4		The Great G...	F. Scott Fitzg...	Classic	Good	Available
5		The Hound of...	Arthur Conan...	Mystery	Good	Available

## Step 6

Code for dq\_checks.sql

```

▶ Run  'final reports.sql'    Book Dataset    Borrower Dataset    Checkout Dataset

1 -- dq_checks.sql
2 -- 1. Check for books with missing titles
3 SELECT * FROM Book
4 WHERE Title IS NULL OR Title = '';
5
6 -- 2. Check for borrowers without phone numbers
7 SELECT * FROM Borrower
8 WHERE BorrowerPhone IS NULL;
9
10 -- 3. Check for checkouts without BorrowerID or BookID
11 SELECT * FROM Checkout
12 WHERE BorrowerID NOT IN (SELECT BorrowerID FROM Borrower)
13     OR BookID NOT IN (SELECT BookID FROM Book);
14
15 -- 4. Check for negative fees or fines
16 SELECT * FROM Borrower
17 WHERE Fees < 0;
18
19 SELECT * FROM Checkout
20 WHERE Fines < 0;
21
22 -- 5. Check for books that are borrowed but not found in the Checkout table
23 SELECT * FROM Book
24 WHERE Status = 'Borrowed'
25     AND BookID NOT IN (SELECT BookID FROM Checkout WHERE ReturnDate IS NULL);
26

```

Return nothing because no missing tiles

BookID	Genre	Title	Author	Condition	Status
1	Fiction	The Great Gatsby	Ernest Hemingway	Good	Available

Return nothing because borrows all have phone numbers

BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees

Return nothing because everyone has valid borrowerID and bookID

#	CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines
1	CK1	BR1	BK1	LB1	2023-01-15	2023-01-01	2023-01-15	\$0.00
2	CK2	BR2	BK2	LB2	2023-01-20	2023-01-05	2023-01-20	\$0.00

Return nothing because no negative fees or fines

BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees

Return nothing because all books that are borrowed are found in the checkout table

#	CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines
1	CK-12345	BR-101	BK-101	LB-101	2023-10-15	2023-09-01	2023-10-15	\$0.00
2	CK-12346	BR-102	BK-102	LB-102	2023-10-20	2023-09-01	2023-10-20	\$0.00

BookID	Genre	Title	Author	Condition	Status
2	Dystopian	1984	George Orwell	Good	Borrowed

## Explain Query Plan before index

The screenshot shows a database interface with several tabs at the top: Run Book Dataset, Borrower Dataset, Checkout Dataset, Librarian Dataset, views.sql, dq\_checks.sql, and Performance Test. Below the tabs, there are two code snippets:

```

1 -- Query 1: Find all borrowed books
2 EXPLAIN QUERY PLAN
3 SELECT * FROM Book WHERE Status = 'Borrowed';
4
5 -- Query 2: Find all checkouts for a certain borrower (example: BorrowerID = 1)
6 EXPLAIN QUERY PLAN
7 SELECT * FROM Checkout WHERE BorrowerID = 1;
8

```

Below the queries is a table with four columns: id, parent, notused, and detail. The data is as follows:

id	parent	notused	detail
2	0	216	SCAN Book

### Explain query plan after index

The screenshot shows a database interface with the same tabs as the previous screenshot. Below the tabs, there are two code snippets:

```

1 -- Query 1: Find all borrowed books
2 EXPLAIN QUERY PLAN
3 SELECT * FROM Book WHERE Status = 'Borrowed';
4
5 -- Query 2: Find all checkouts for a certain borrower (example: BorrowerID = 1)
6 EXPLAIN QUERY PLAN
7 SELECT * FROM Checkout WHERE BorrowerID = 1;
8 -- Create indexes to improve performance
9 CREATE INDEX idx_book_status ON Book(Status);
10 CREATE INDEX idx_checkout_borrowerid ON Checkout(BorrowerID);
11

```

Below the queries is a table with four columns: id, parent, notused, and detail. The data is as follows:

id	parent	notused	detail
3	0	62	SEARCH Book USING INDEX idx_book_status (Stat...

[performance.md](#)

```
▶ Run 'final reports.sql' Book Dataset Borrower Dataset Checkout Dataset Librarian Datset vi

1 -- performance.md
2
3 The EXPLAIN QUERY PLAN shows how SQLite runs a QUERY.
4 It shows us IF SQLite IS scanning the whole TABLE OR USING an INDEX.
5
6 BEFORE adding INDEXES
7 - FOR `SELECT * FROM Book WHERE Status = 'Borrowed'`, SQLite showed:
8
9 SCAN Book
10
11 This means it checks every ROW IN the Book TABLE TO find matches.
12
13 - FOR `SELECT * FROM Checkout WHERE BorrowerID = 1`, SQLite showed:
14
15 Scan Checkout
16
17 This also checks every ROW IN the TABLE.
18
19 AFTER creating these INDEXES:
20 sql
21 CREATE INDEX idx_book_status ON Book(Status);
22 CREATE INDEX idx_checkout_borrowerid ON Checkout(BorrowerID);
23
24 SEARCH Book USING INDEX idx_book_status
25 SEARCH Checkout USING INDEX idx_checkout_borrowerid
26
27 Uses rows it needs INSTEAD OF the whole TABLE which make queries run faster
28
29 Other notes
30
31 I also edited the imported sheets. I took out the c1, c2, etc... because it was confusing FOR me AND sqliteonline.
32 The code that I was programming did NOT respond the way that I wanted it TO.
33 I took out the ROW WITH ALL the labels AND put the labels INTO the spots WHERE c1, c2, etc... were.
34 This way sqliteonline READ this AS my COLUMN INSTEAD OF telling me the COLUMN didn't exist.
35
```