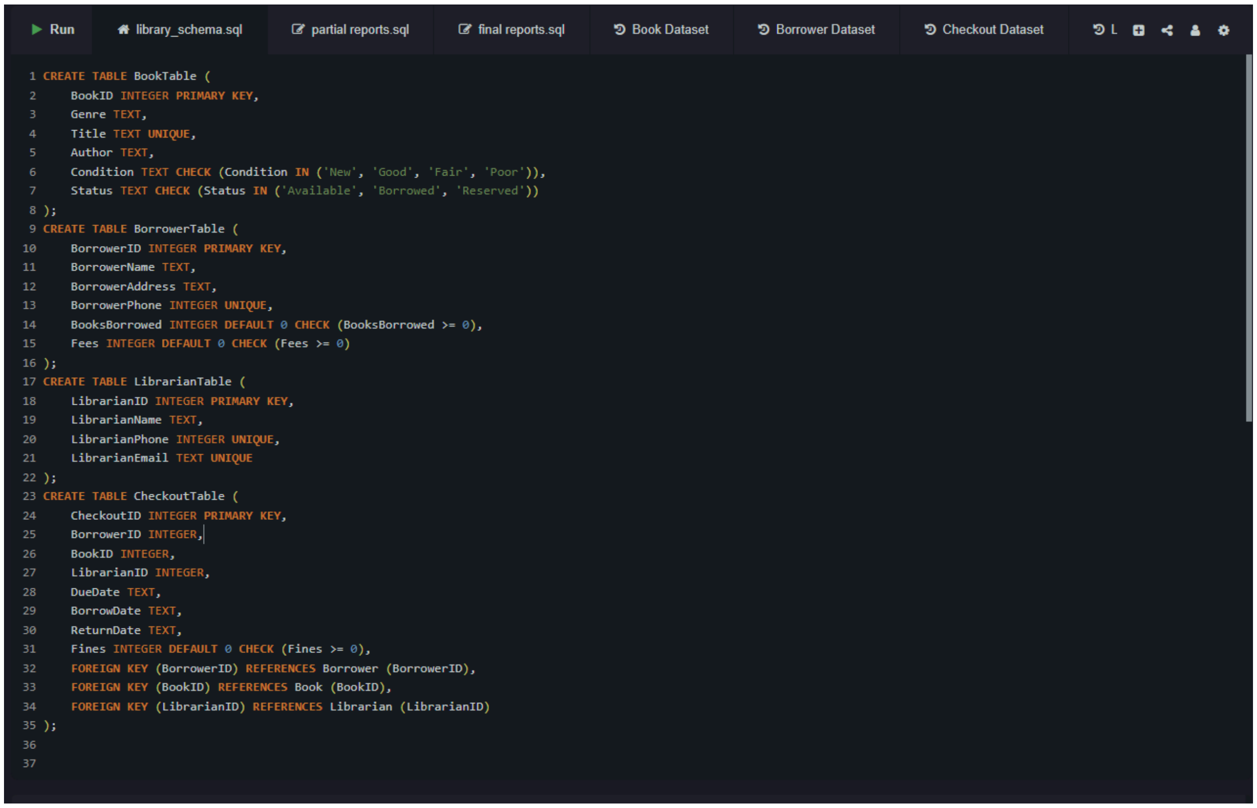
Step 1

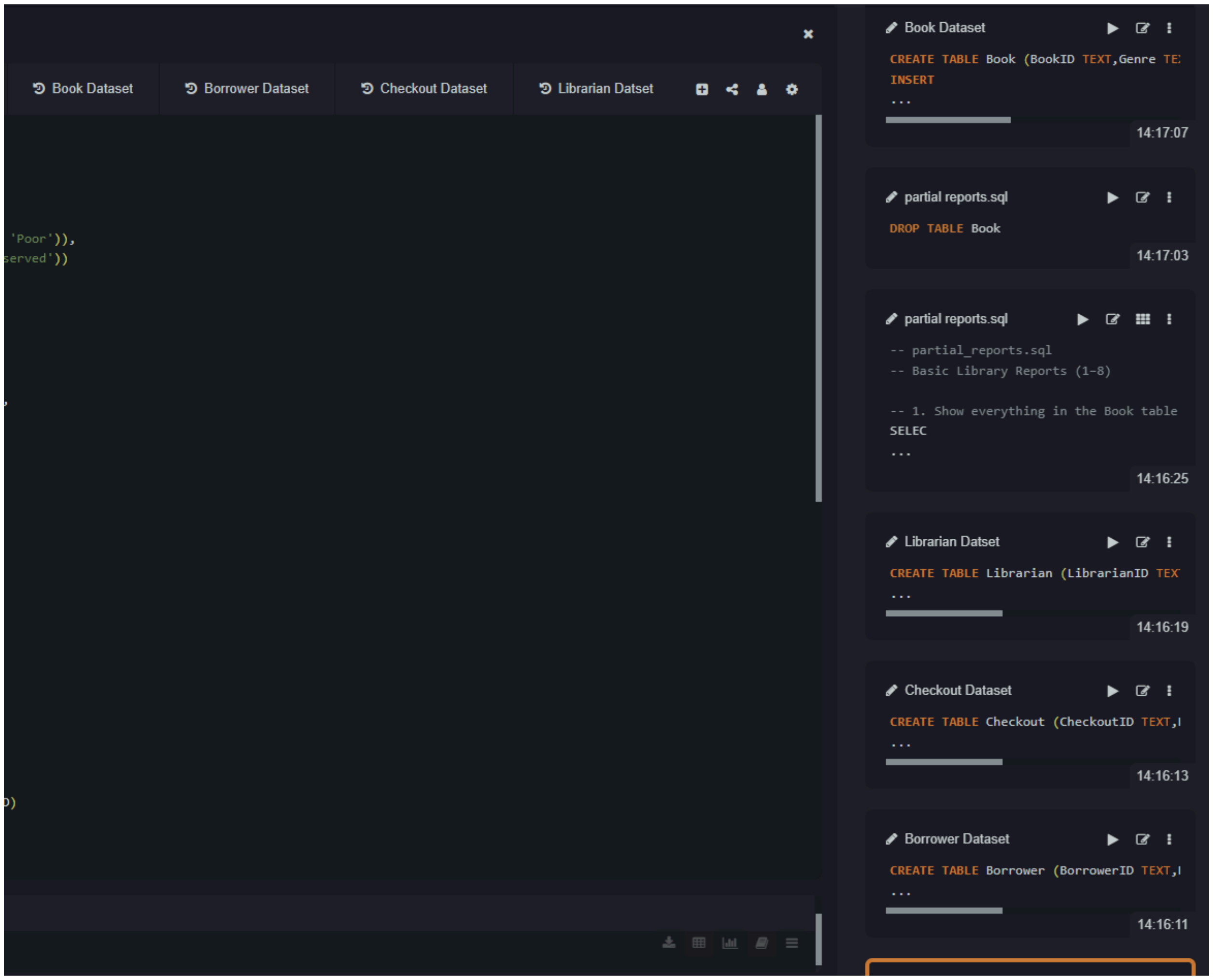
https://dbdiagram.io/d/Project-2-ER-Diagram-Jeffery-Yong-68b9cdec61a46d388e81e634

Step 2
SS of code -



```sql
CREATE TABLE BookTable (
    BookID INTEGER PRIMARY KEY,
    Genre TEXT,
    Title TEXT UNIQUE,
    Author TEXT,
    Condition TEXT CHECK (Condition IN ('New', 'Good', 'Fair', 'Poor')),
    Status TEXT CHECK (Status IN ('Available', 'Borrowed', 'Reserved'))
);
CREATE TABLE BorrowerTable (
    BorrowerID INTEGER PRIMARY KEY,
    BorrowerName TEXT,
    BorrowerAddress TEXT,
    BorrowerPhone INTEGER UNIQUE,
    BooksBorrowed INTEGER DEFAULT 0 CHECK (BooksBorrowed >= 0),
    Fees INTEGER DEFAULT 0 CHECK (Fees >= 0)
);
CREATE TABLE LibrarianTable (
    LibrarianID INTEGER PRIMARY KEY,
    LibrarianName TEXT,
    LibrarianPhone INTEGER UNIQUE,
    LibrarianEmail TEXT UNIQUE
);
CREATE TABLE CheckoutTable (
    CheckoutID INTEGER PRIMARY KEY,
    BorrowerID INTEGER,
    BookID INTEGER,
    LibrarianID INTEGER,
    DueDate TEXT,
    BorrowDate TEXT,
    ReturnDate TEXT,
    Fines INTEGER DEFAULT 0 CHECK (Fines >= 0),
    FOREIGN KEY (BorrowerID) REFERENCES Borrower (BorrowerID),
    FOREIGN KEY (BookID) REFERENCES Book (BookID),
    FOREIGN KEY (LibrarianID) REFERENCES Librarian (LibrarianID)
);
```

Step 3
SS of import -



```
'Poor')),
served'))
```

```
)
```

Book Dataset 14:17:07
```
CREATE TABLE Book (BookID TEXT,Genre TE
INSERT
...
```

partial reports.sql 14:17:03
```
DROP TABLE Book
```

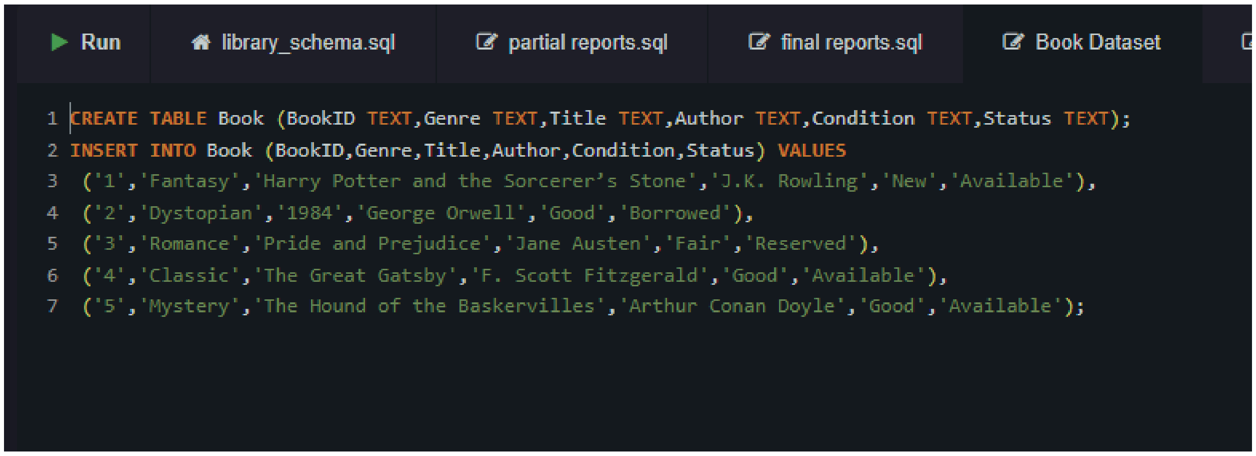
partial reports.sql 14:16:25
```
-- partial_reports.sql
-- Basic Library Reports (1-8)

-- 1. Show everything in the Book table
SELEC
...
```

Librarian Datset 14:16:19
```
CREATE TABLE Librarian (LibrarianID TEX
...
```

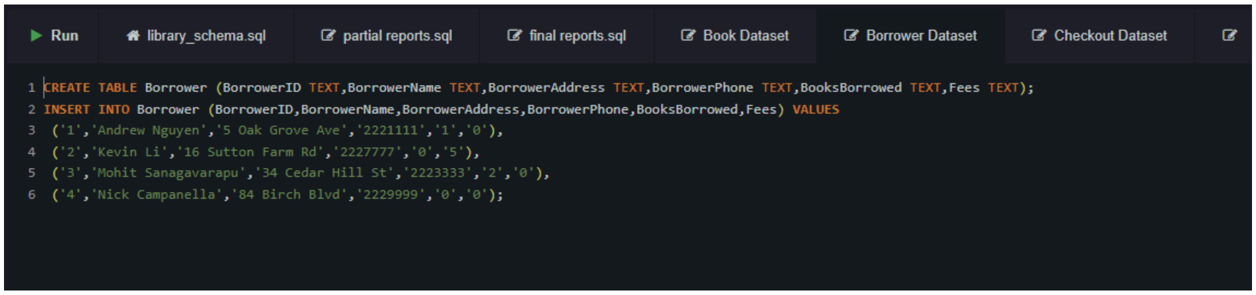
Checkout Dataset 14:16:13
```
CREATE TABLE Checkout (CheckoutID TEXT,I
...
```

Borrower Dataset 14:16:11
```
CREATE TABLE Borrower (BorrowerID TEXT,I
...
```

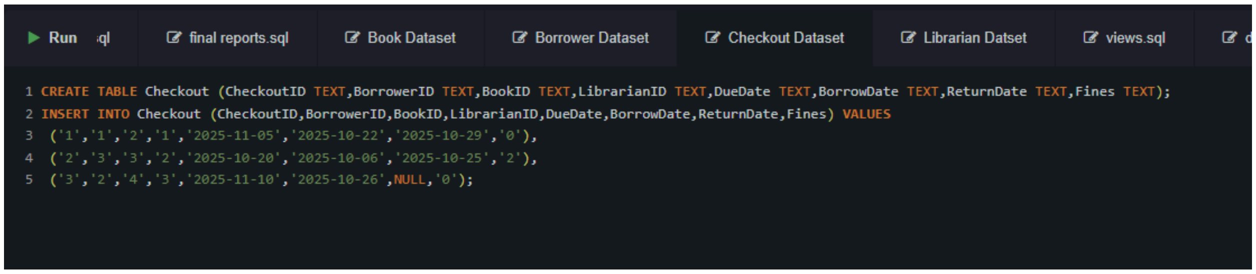
Imported Datasets



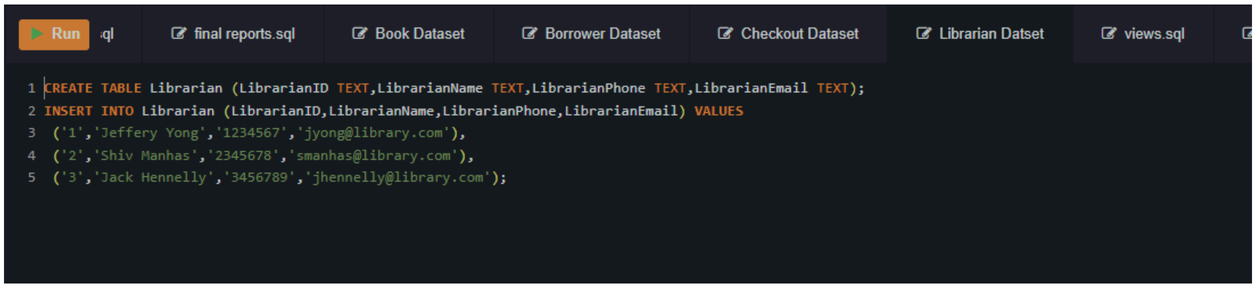
```
CREATE TABLE Book (BookID TEXT,Genre TEXT,Title TEXT,Author TEXT,Condition TEXT,Status TEXT);
INSERT INTO Book (BookID,Genre,Title,Author,Condition,Status) VALUES
 ('1','Fantasy','Harry Potter and the Sorcerer's Stone','J.K. Rowling','New','Available'),
 ('2','Dystopian','1984','George Orwell','Good','Borrowed'),
 ('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');
```



```sql
CREATE TABLE Borrower (BorrowerID TEXT,BorrowerName TEXT,BorrowerAddress TEXT,BorrowerPhone TEXT,BooksBorrowed TEXT,Fees TEXT);
INSERT INTO Borrower (BorrowerID,BorrowerName,BorrowerAddress,BorrowerPhone,BooksBorrowed,Fees) VALUES
 ('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');
```

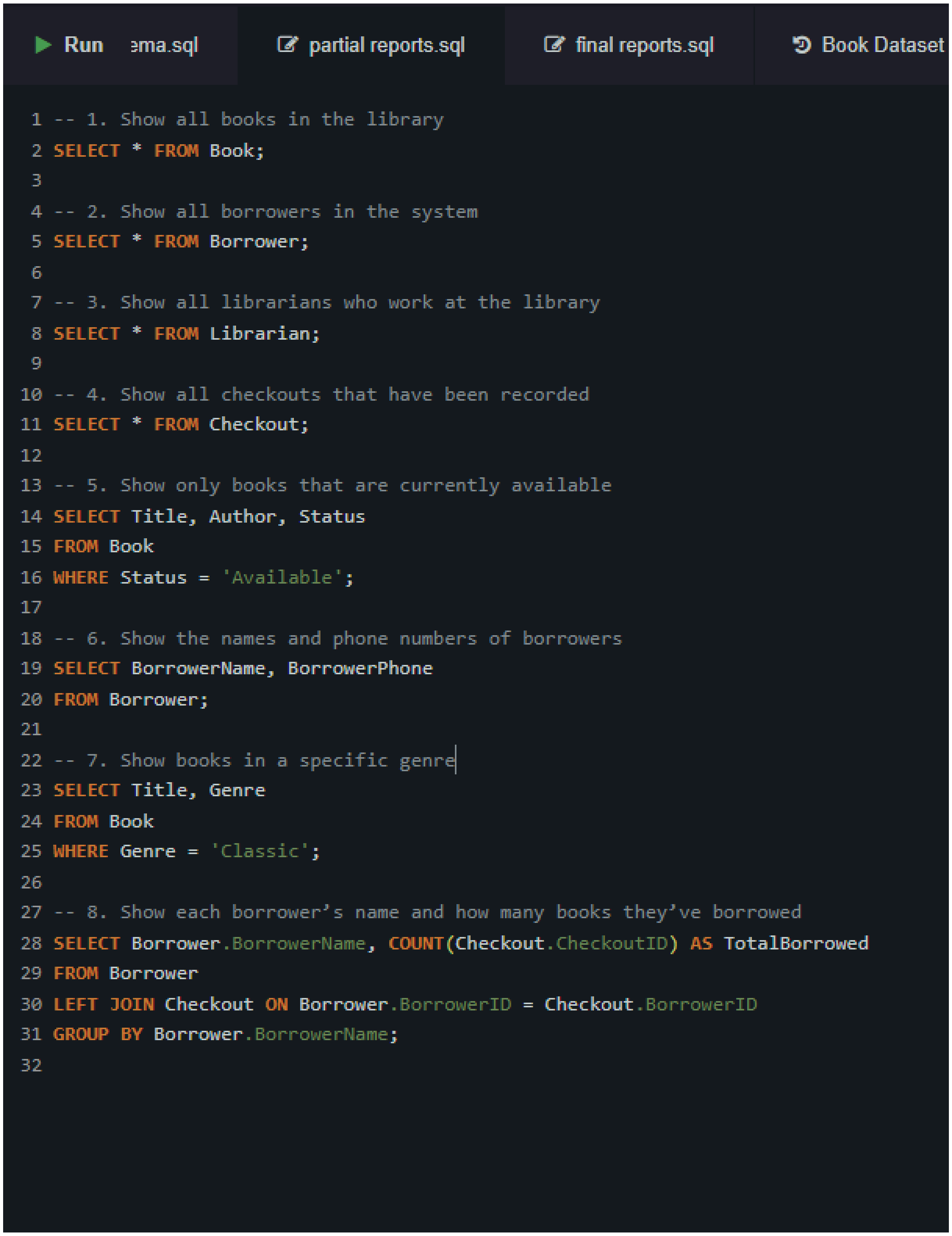


```sql
CREATE TABLE Checkout (CheckoutID TEXT,BorrowerID TEXT,BookID TEXT,LibrarianID TEXT,DueDate TEXT,BorrowDate TEXT,ReturnDate TEXT,Fines TEXT);
INSERT INTO Checkout (CheckoutID,BorrowerID,BookID,LibrarianID,DueDate,BorrowDate,ReturnDate,Fines) VALUES
 ('1','1','2','1','2025-11-05','2025-10-22','2025-10-29','0'),
 ('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');
```



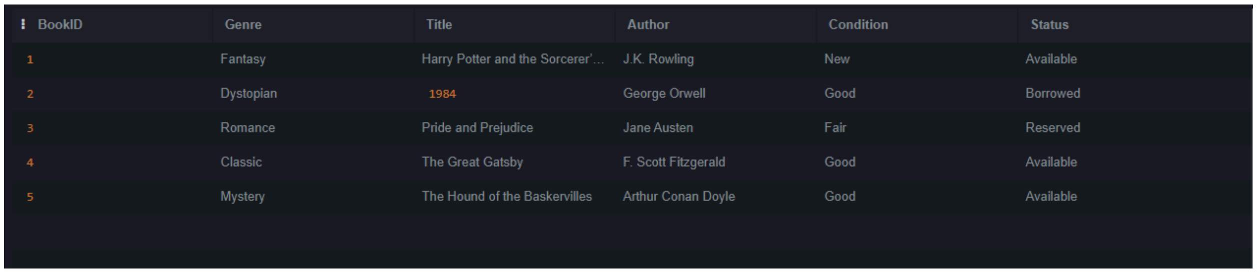
```sql
CREATE TABLE Librarian (LibrarianID TEXT,LibrarianName TEXT,LibrarianPhone TEXT,LibrarianEmail TEXT);
INSERT INTO Librarian (LibrarianID,LibrarianName,LibrarianPhone,LibrarianEmail) VALUES
 ('1','Jeffery Yong','1234567','jyong@library.com'),
 ('2','Shiv Manhas','2345678','smanhas@library.com'),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

Step 4
Code -


Run ema.sql | partial reports.sql | final reports.sql | Book Dataset

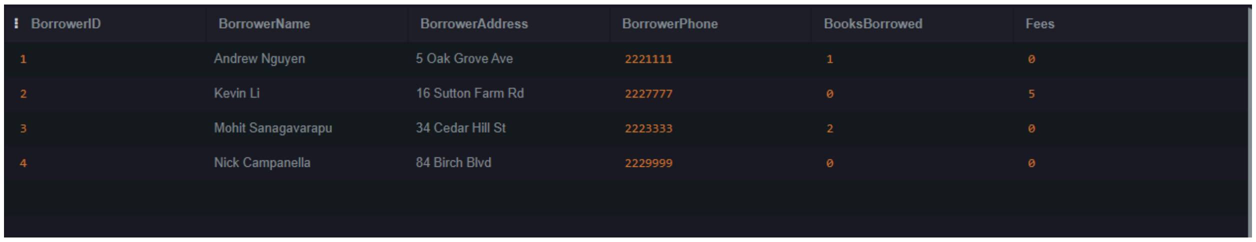
```sql
-- 1. Show all books in the library
SELECT * FROM Book;

-- 2. Show all borrowers in the system
SELECT * FROM Borrower;

-- 3. Show all librarians who work at the library
SELECT * FROM Librarian;

-- 4. Show all checkouts that have been recorded
SELECT * FROM Checkout;

-- 5. Show only books that are currently available
SELECT Title, Author, Status
FROM Book
WHERE Status = 'Available';

-- 6. Show the names and phone numbers of borrowers
SELECT BorrowerName, BorrowerPhone
FROM Borrower;

-- 7. Show books in a specific genre
SELECT Title, Genre
FROM Book
WHERE Genre = 'Classic';

-- 8. Show each borrower’s name and how many books they’ve borrowed
SELECT Borrower.BorrowerName, COUNT(Checkout.CheckoutID) AS TotalBorrowed
FROM Borrower
LEFT JOIN Checkout ON Borrower.BorrowerID = Checkout.BorrowerID
GROUP BY Borrower.BorrowerName;
```

1-



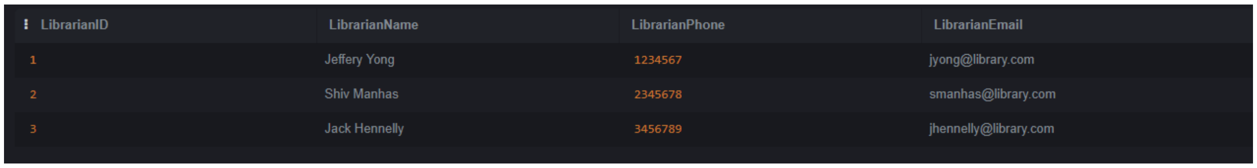
BookID	Genre	Title	Author	Condition	Status
1	Fantasy	Harry Potter and the Sorcerer'...	J.K. Rowling	New	Available
2	Dystopian	1984	George Orwell	Good	Borrowed
3	Romance	Pride and Prejudice	Jane Austen	Fair	Reserved
4	Classic	The Great Gatsby	F. Scott Fitzgerald	Good	Available
5	Mystery	The Hound of the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

2-



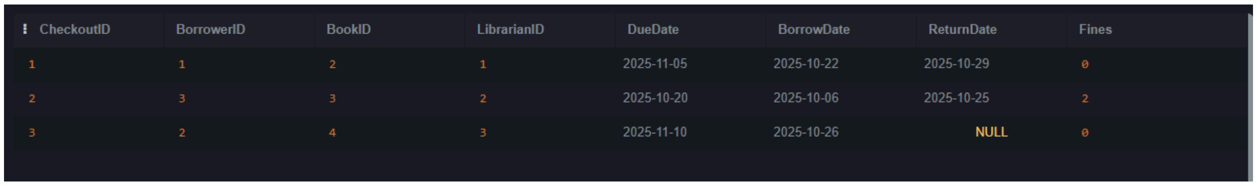
BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees
1	Andrew Nguyen	5 Oak Grove Ave	2221111	1	0
2	Kevin Li	16 Sutton Farm Rd	2227777	0	5
3	Mohit Sanagavarapu	34 Cedar Hill St	2223333	2	0
4	Nick Campanella	84 Birch Blvd	2229999	0	0

3-



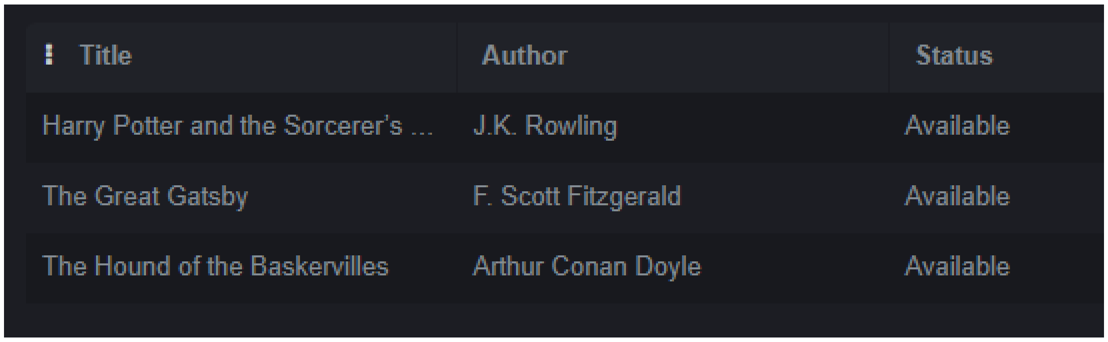
LibrarianID	LibrarianName	LibrarianPhone	LibrarianEmail
1	Jeffery Yong	1234567	jyong@library.com
2	Shiv Manhas	2345678	smanhas@library.com
3	Jack Hennelly	3456789	jhennelly@library.com

4-



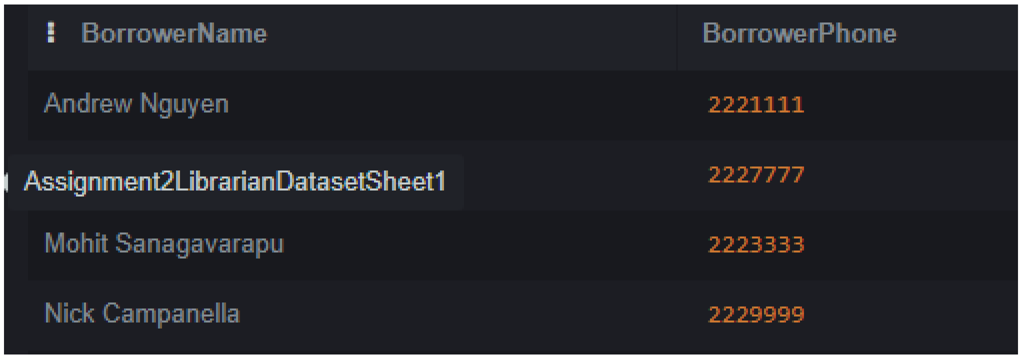
CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines
1	1	2	1	2025-11-05	2025-10-22	2025-10-29	0
2	3	3	2	2025-10-20	2025-10-06	2025-10-25	2
3	2	4	3	2025-11-10	2025-10-26	NULL	0

5-



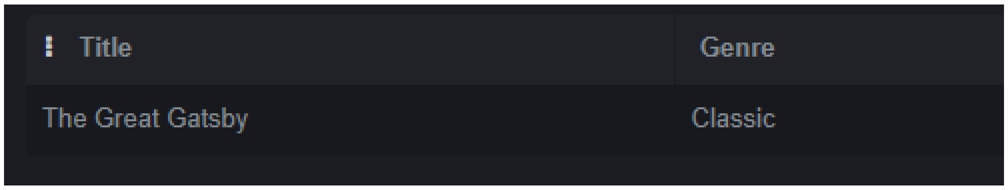
Title	Author	Status
Harry Potter and the Sorcerer's ...	J.K. Rowling	Available
The Great Gatsby	F. Scott Fitzgerald	Available
The Hound of the Baskervilles	Arthur Conan Doyle	Available

6-



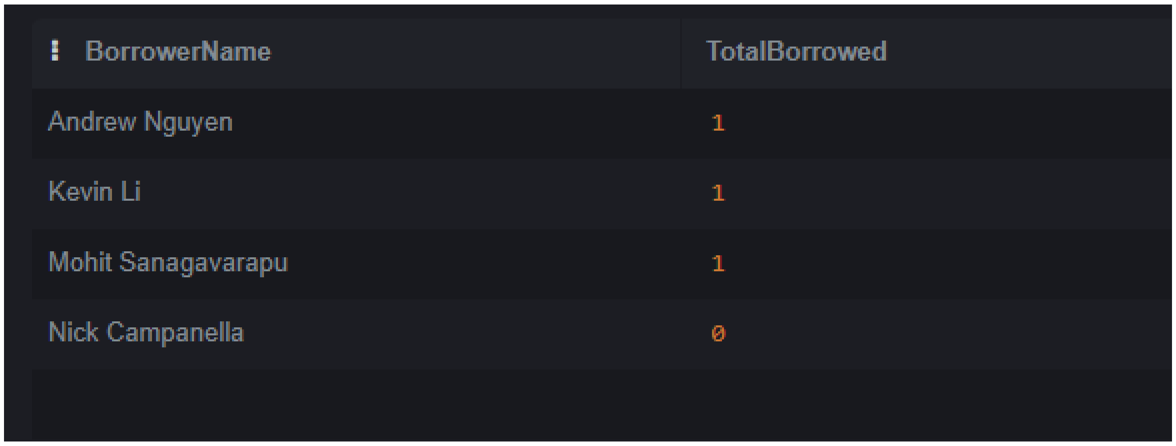
BorrowerName	BorrowerPhone
Andrew Nguyen	2221111
Assignment2LibrarianDatasetSheet1	2227777
Mohit Sanagavarapu	2223333
Nick Campanella	2229999

7-



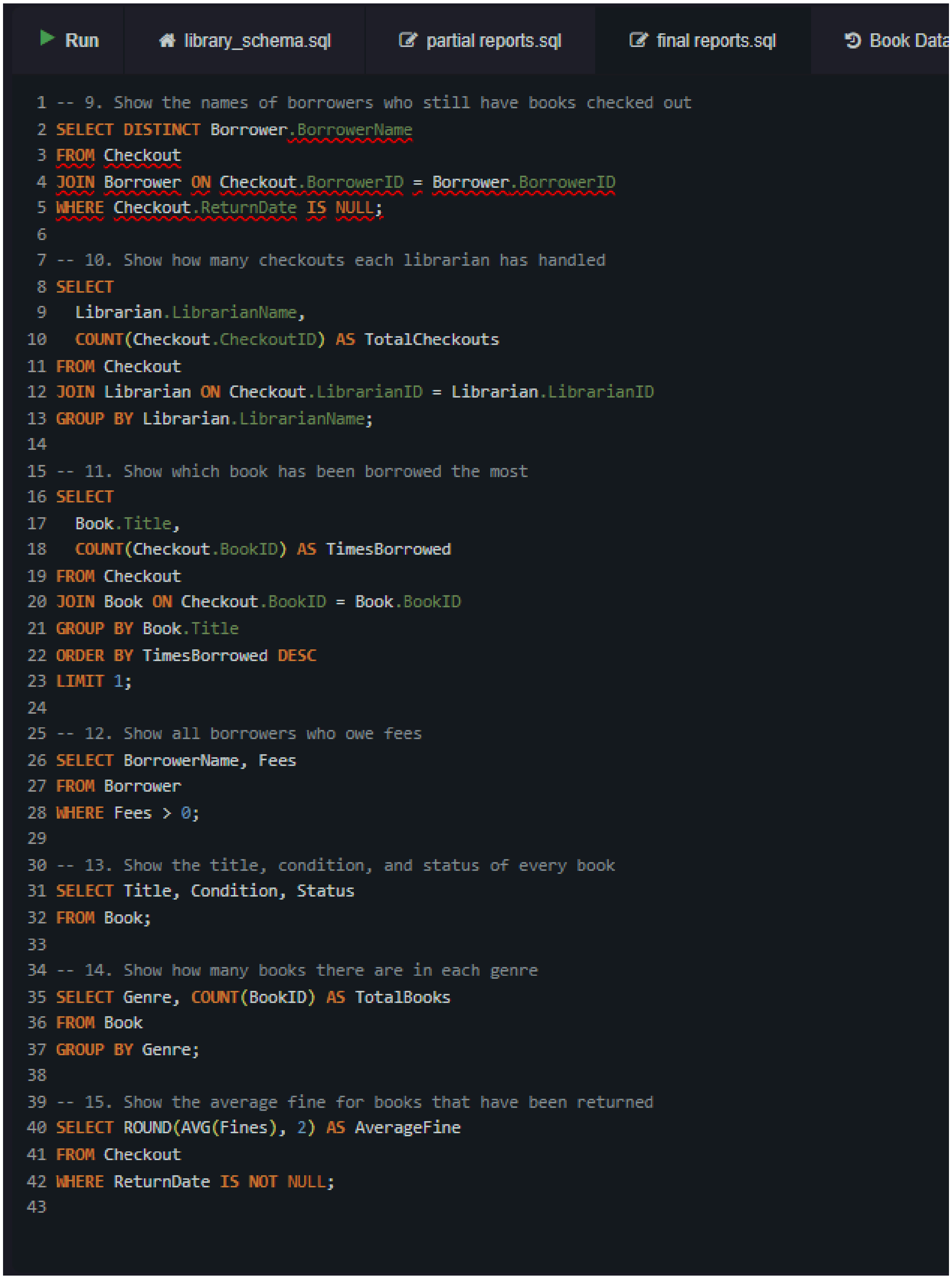
Title	Genre
The Great Gatsby	Classic

8-



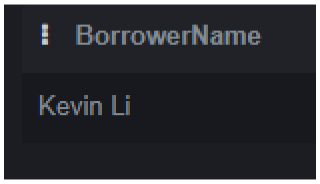
BorrowerName	TotalBorrowed
Andrew Nguyen	1
Kevin Li	1
Mohit Sanagavarapu	1
Nick Campanella	0

Step 5
Code -



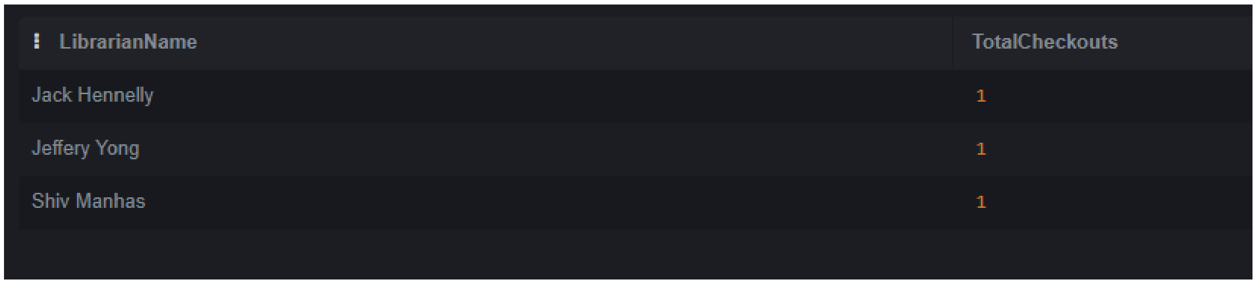
```sql
-- 9. Show the names of borrowers who still have books checked out
SELECT DISTINCT Borrower.BorrowerName
FROM Checkout
JOIN Borrower ON Checkout.BorrowerID = Borrower.BorrowerID
WHERE Checkout.ReturnDate IS NULL;

-- 10. Show how many checkouts each librarian has handled
SELECT
  Librarian.LibrarianName,
  COUNT(Checkout.CheckoutID) AS TotalCheckouts
FROM Checkout
JOIN Librarian ON Checkout.LibrarianID = Librarian.LibrarianID
GROUP BY Librarian.LibrarianName;

-- 11. Show which book has been borrowed the most
SELECT
  Book.Title,
  COUNT(Checkout.BookID) AS TimesBorrowed
FROM Checkout
JOIN Book ON Checkout.BookID = Book.BookID
GROUP BY Book.Title
ORDER BY TimesBorrowed DESC
LIMIT 1;

-- 12. Show all borrowers who owe fees
SELECT BorrowerName, Fees
FROM Borrower
WHERE Fees > 0;

-- 13. Show the title, condition, and status of every book
SELECT Title, Condition, Status
FROM Book;

-- 14. Show how many books there are in each genre
SELECT Genre, COUNT(BookID) AS TotalBooks
FROM Book
GROUP BY Genre;

-- 15. Show the average fine for books that have been returned
SELECT ROUND(AVG(Fines), 2) AS AverageFine
FROM Checkout
WHERE ReturnDate IS NOT NULL;
```

9-



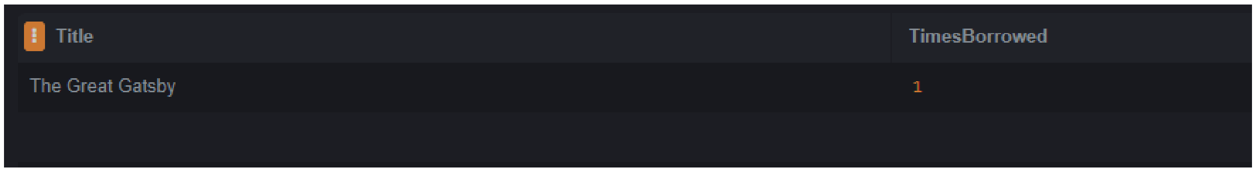
BorrowerName
Kevin Li

10-



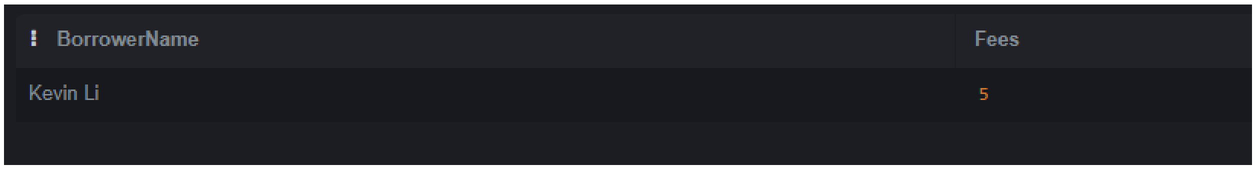
LibrarianName	TotalCheckouts
Jack Hennelly	1
Jeffery Yong	1
Shiv Manhas	1

11-



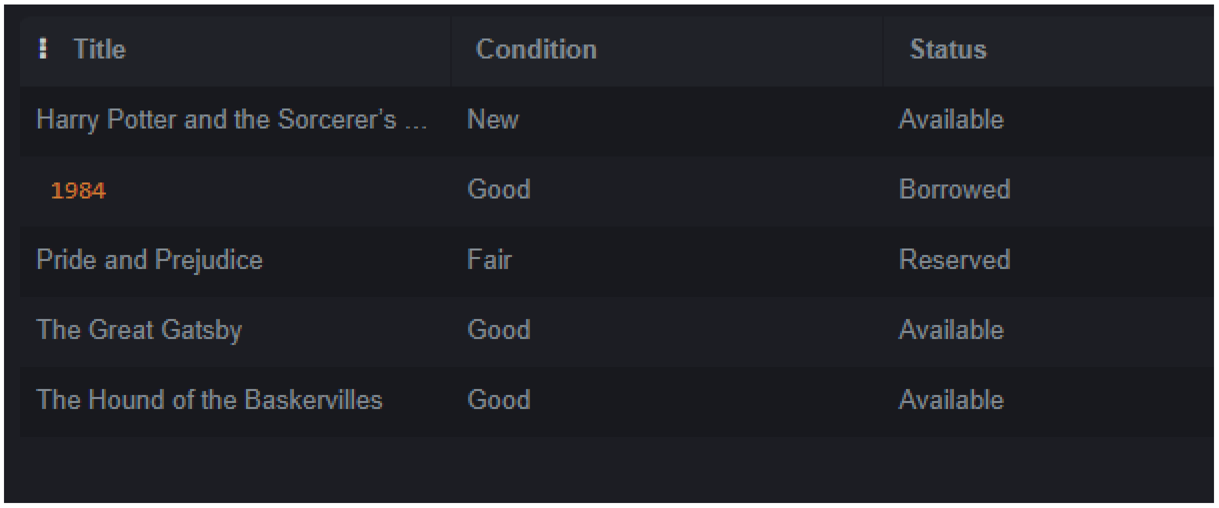
Title	TimesBorrowed
The Great Gatsby	1

12-



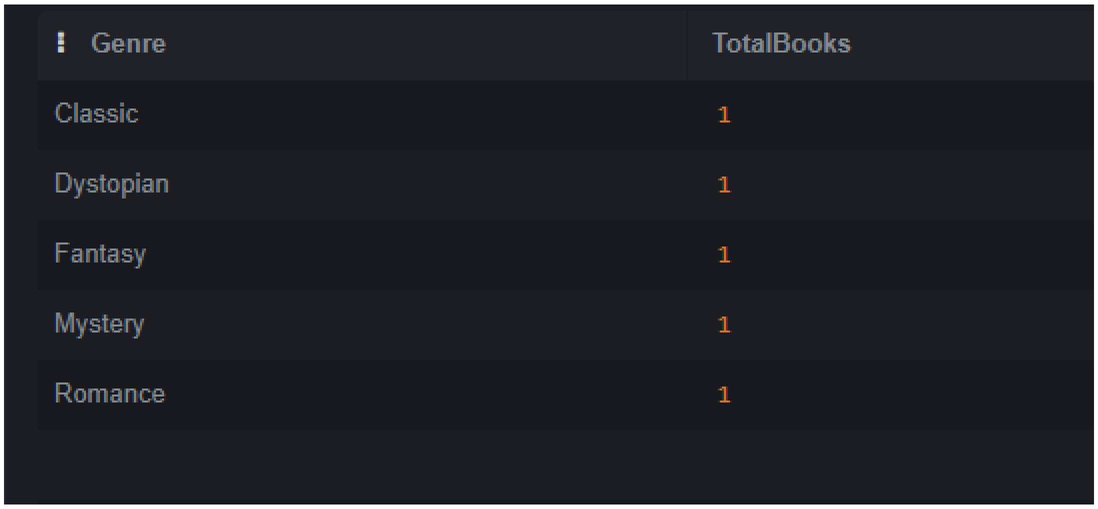
BorrowerName	Fees
Kevin Li	5

13-



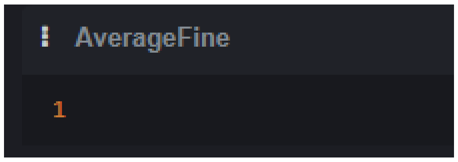
Title	Condition	Status
Harry Potter and the Sorcerer's ...	New	Available
1984	Good	Borrowed
Pride and Prejudice	Fair	Reserved
The Great Gatsby	Good	Available
The Hound of the Baskervilles	Good	Available

14-



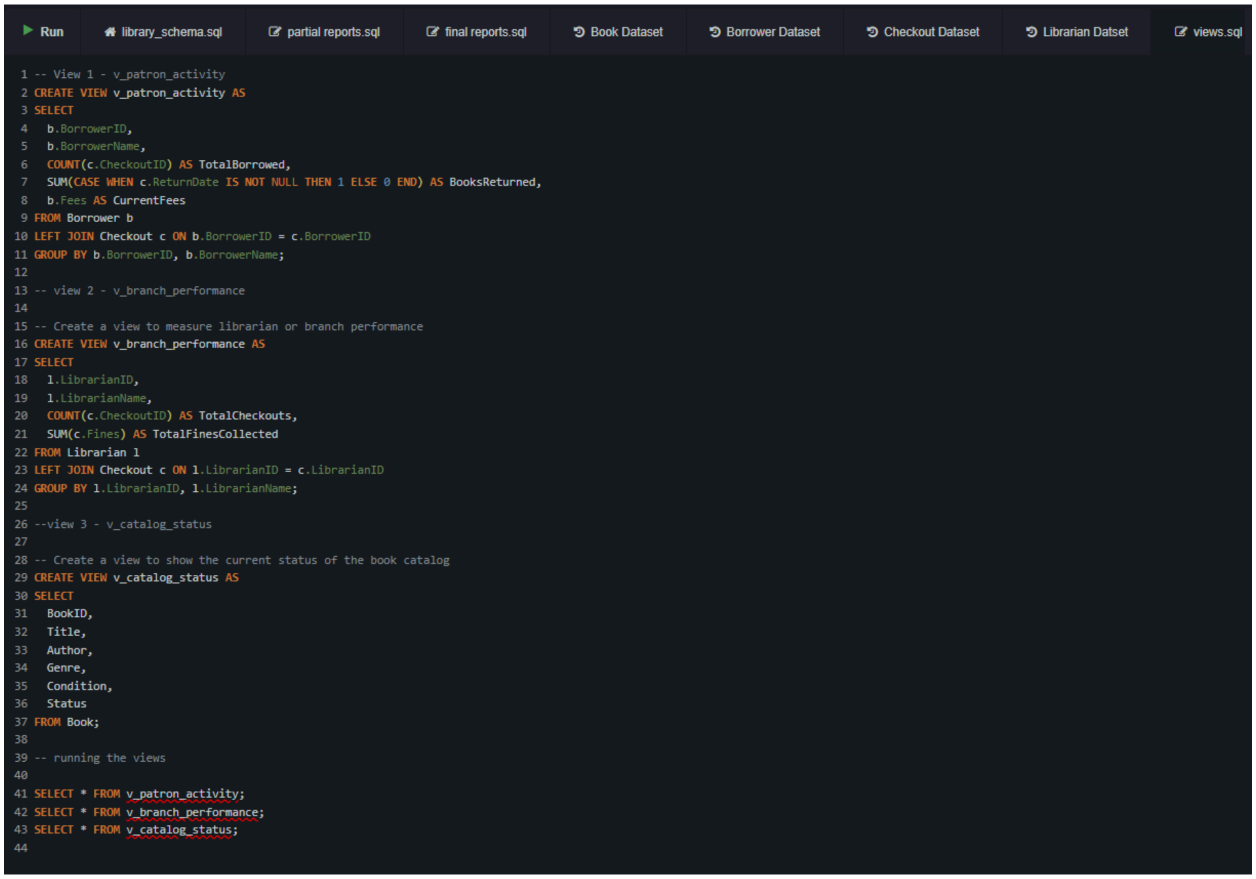
Genre	TotalBooks
Classic	1
Dystopian	1
Fantasy	1
Mystery	1
Romance	1

15-



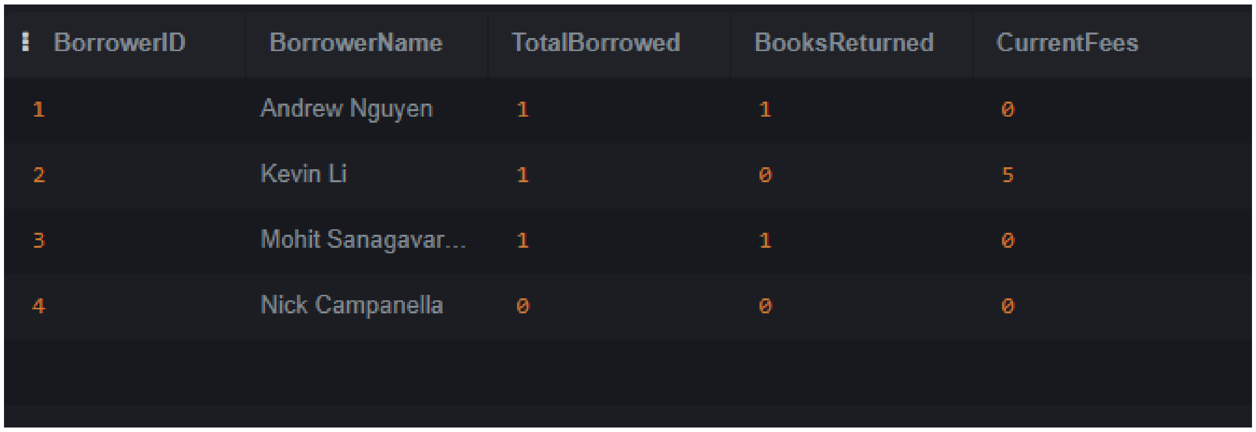
AverageFine
1

Views.sql
Code -



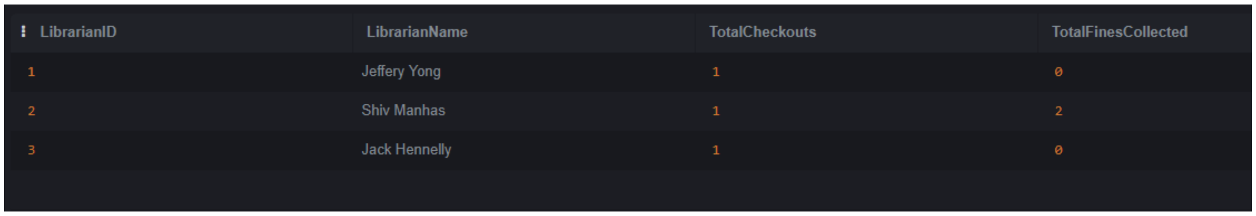
```sql
-- View 1 - v_patron_activity
CREATE VIEW v_patron_activity AS
SELECT
  b.BorrowerID,
  b.BorrowerName,
  COUNT(c.CheckoutID) AS TotalBorrowed,
  SUM(CASE WHEN c.ReturnDate IS NOT NULL THEN 1 ELSE 0 END) AS BooksReturned,
  b.Fees AS CurrentFees
FROM Borrower b
LEFT JOIN Checkout c ON b.BorrowerID = c.BorrowerID
GROUP BY b.BorrowerID, b.BorrowerName;

-- view 2 - v_branch_performance

-- Create a view to measure librarian or branch performance
CREATE VIEW v_branch_performance AS
SELECT
  l.LibrarianID,
  l.LibrarianName,
  COUNT(c.CheckoutID) AS TotalCheckouts,
  SUM(c.Fines) AS TotalFinesCollected
FROM Librarian l
LEFT JOIN Checkout c ON l.LibrarianID = c.LibrarianID
GROUP BY l.LibrarianID, l.LibrarianName;

--view 3 - v_catalog_status

-- Create a view to show the current status of the book catalog
CREATE VIEW v_catalog_status AS
SELECT
  BookID,
  Title,
  Author,
  Genre,
  Condition,
  Status
FROM Book;

-- running the views

SELECT * FROM v_patron_activity;
SELECT * FROM v_branch_performance;
SELECT * FROM v_catalog_status;
```

v_patron_activity



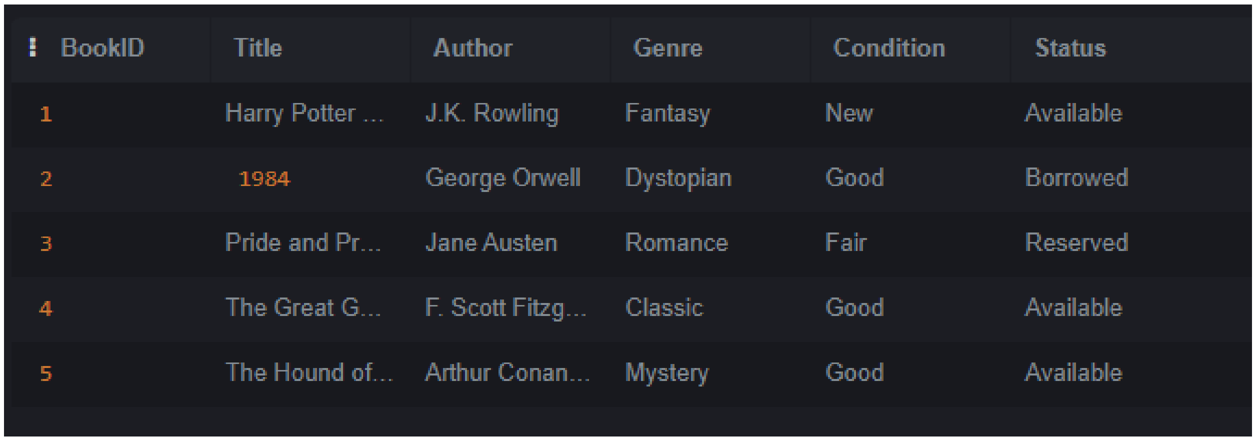
BorrowerID	BorrowerName	TotalBorrowed	BooksReturned	CurrentFees
1	Andrew Nguyen	1	1	0
2	Kevin Li	1	0	5
3	Mohit Sanagavar...	1	1	0
4	Nick Campanella	0	0	0

v_branch_performance



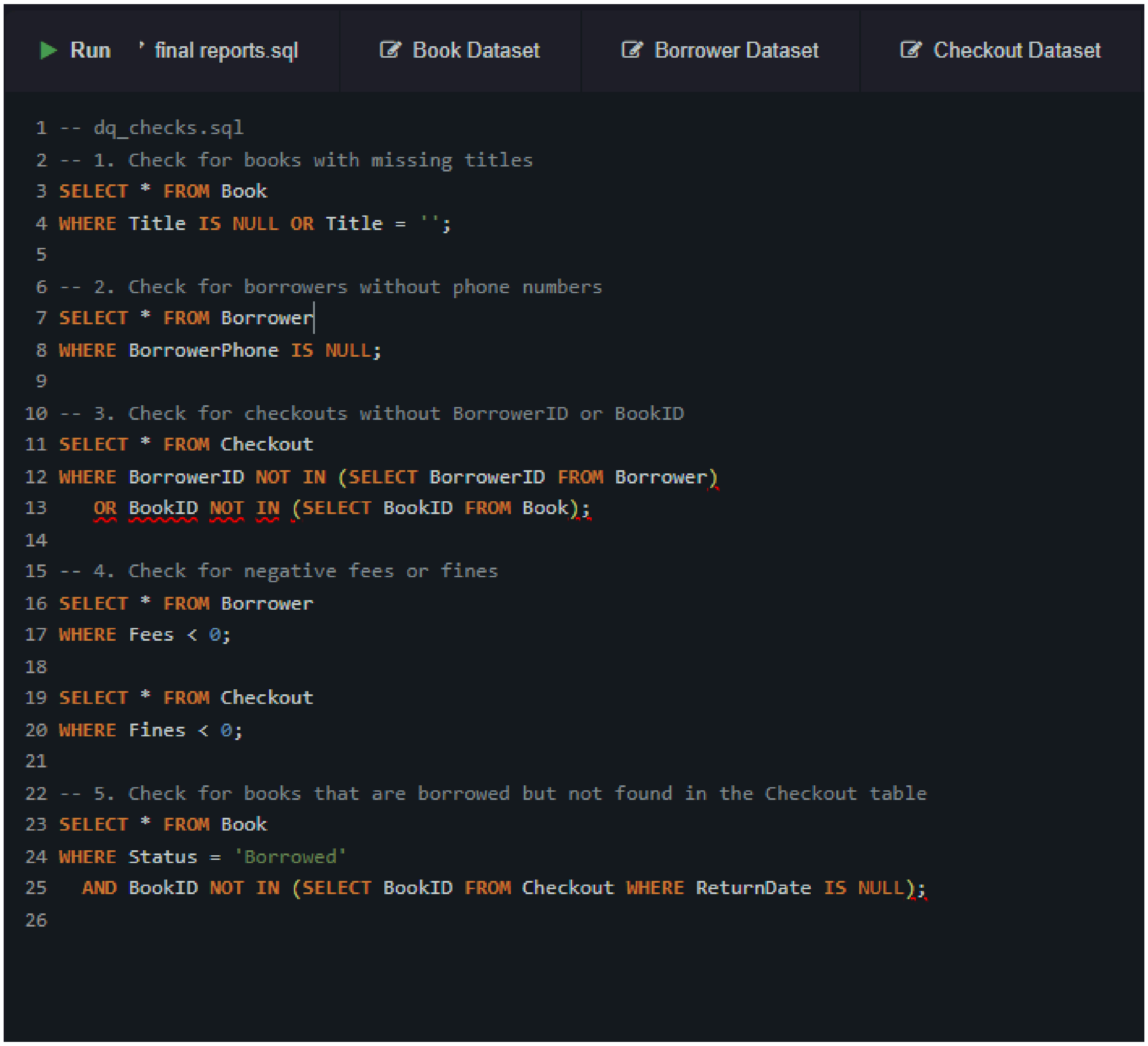
LibrarianID	LibrarianName	TotalCheckouts	TotalFinesCollected
1	Jeffery Yong	1	0
2	Shiv Manhas	1	2
3	Jack Hennelly	1	0

v_catalog_status



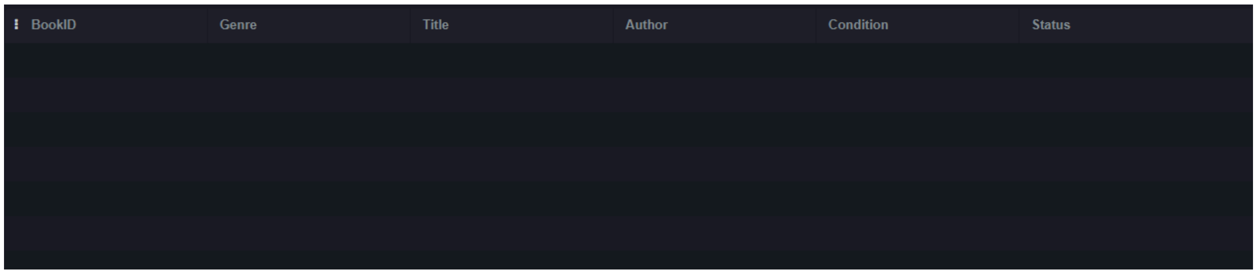
BookID	Title	Author	Genre	Condition	Status
1	Harry Potter ...	J.K. Rowling	Fantasy	New	Available
2	1984	George Orwell	Dystopian	Good	Borrowed
3	Pride and Pr...	Jane Austen	Romance	Fair	Reserved
4	The Great G...	F. Scott Fitzg...	Classic	Good	Available
5	The Hound of...	Arthur Conan...	Mystery	Good	Available

Step 6
Code for dq_checks.sql



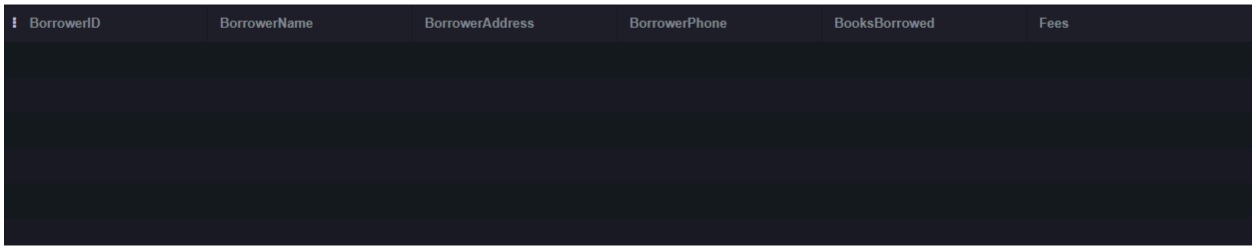
```sql
-- dq_checks.sql
-- 1. Check for books with missing titles
SELECT * FROM Book
WHERE Title IS NULL OR Title = '';

-- 2. Check for borrowers without phone numbers
SELECT * FROM Borrower
WHERE BorrowerPhone IS NULL;

-- 3. Check for checkouts without BorrowerID or BookID
SELECT * FROM Checkout
WHERE BorrowerID NOT IN (SELECT BorrowerID FROM Borrower)
   OR BookID NOT IN (SELECT BookID FROM Book);

-- 4. Check for negative fees or fines
SELECT * FROM Borrower
WHERE Fees < 0;

SELECT * FROM Checkout
WHERE Fines < 0;

-- 5. Check for books that are borrowed but not found in the Checkout table
SELECT * FROM Book
WHERE Status = 'Borrowed'
  AND BookID NOT IN (SELECT BookID FROM Checkout WHERE ReturnDate IS NULL);

```

Return nothing because no missing tiles



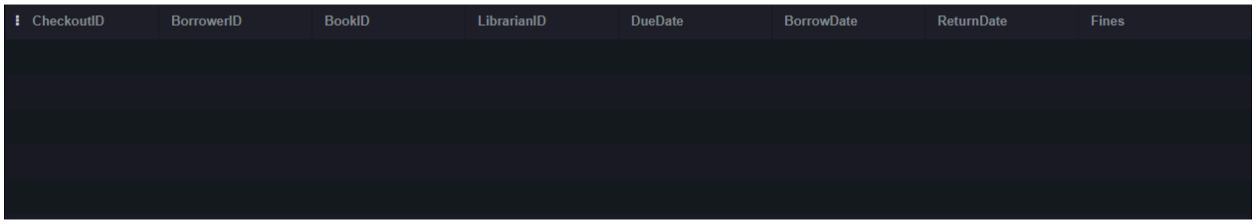
BookID	Genre	Title	Author	Condition	Status

Return nothing because borrows all have phone numbers



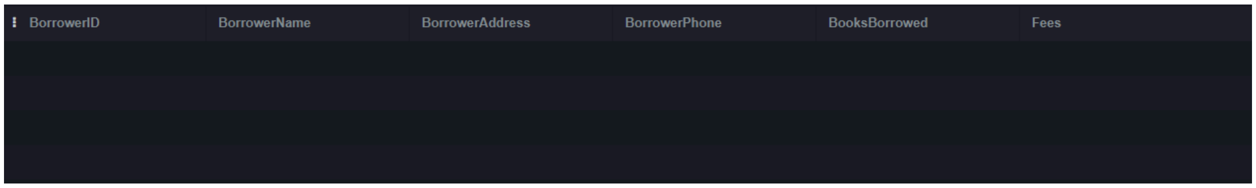
BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees

Return nothing because everyone has valid borrowerID and bookID



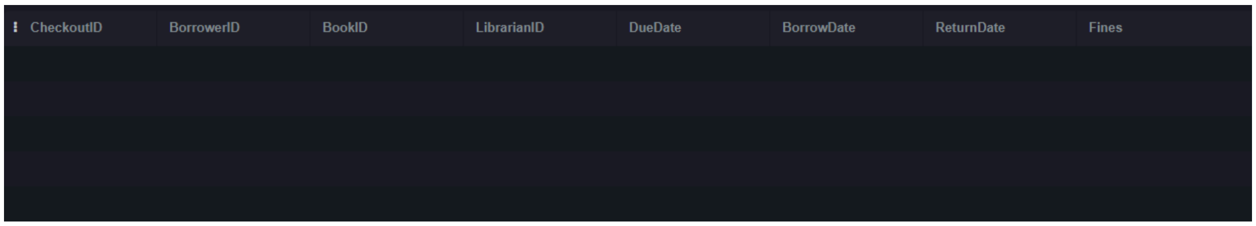
CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines

Return nothing because no negative fees or fines

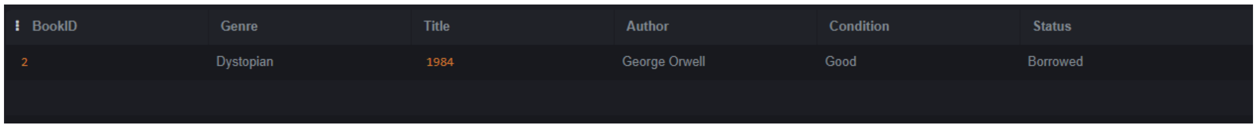


BorrowerID	BorrowerName	BorrowerAddress	BorrowerPhone	BooksBorrowed	Fees

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

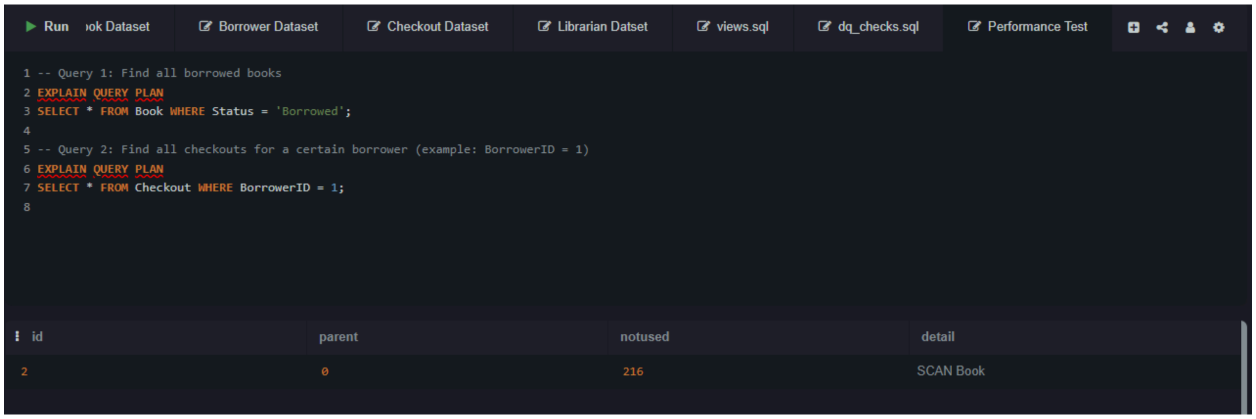


CheckoutID	BorrowerID	BookID	LibrarianID	DueDate	BorrowDate	ReturnDate	Fines



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

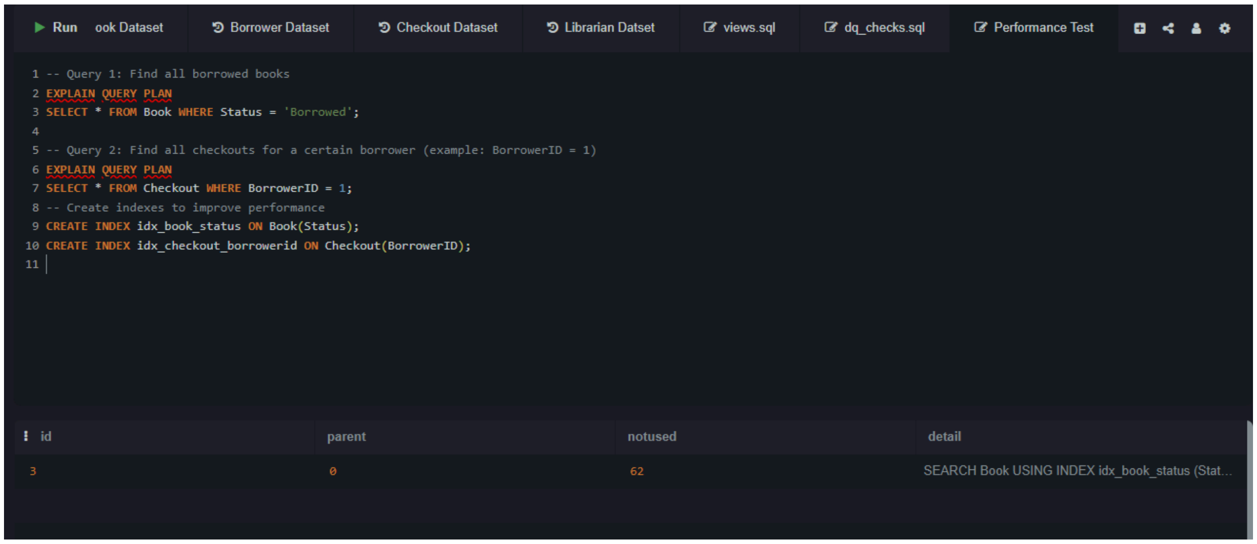
Explain Query Plan before index



```
-- Query 1: Find all borrowed books
EXPLAIN QUERY PLAN
SELECT * FROM Book WHERE Status = 'Borrowed';

-- Query 2: Find all checkouts for a certain borrower (example: BorrowerID = 1)
EXPLAIN QUERY PLAN
SELECT * FROM Checkout WHERE BorrowerID = 1;
```

id	parent	notused	detail
2	0	216	SCAN Book

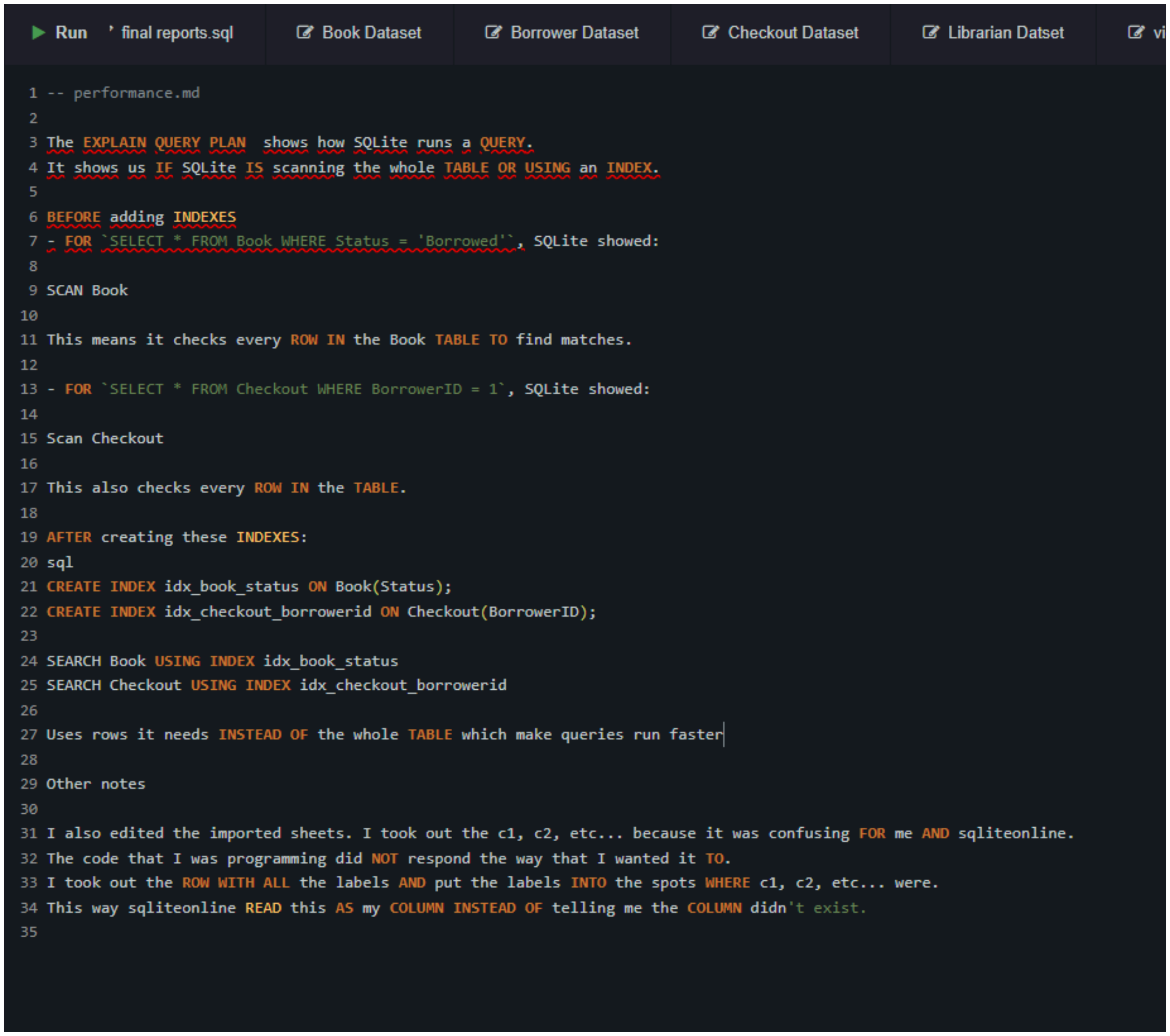
Explain query plan after index



```
-- Query 1: Find all borrowed books
EXPLAIN QUERY PLAN
SELECT * FROM Book WHERE Status = 'Borrowed';

-- Query 2: Find all checkouts for a certain borrower (example: BorrowerID = 1)
EXPLAIN QUERY PLAN
SELECT * FROM Checkout WHERE BorrowerID = 1;
-- Create indexes to improve performance
CREATE INDEX idx_book_status ON Book(Status);
CREATE INDEX idx_checkout_borrowerid ON Checkout(BorrowerID);
```

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

performance.md



```
-- performance.md

The EXPLAIN QUERY PLAN  shows how SQLite runs a QUERY.
It shows us IF SQLite IS scanning the whole TABLE OR USING an INDEX.

BEFORE adding INDEXES
- FOR `SELECT * FROM Book WHERE Status = 'Borrowed'`, SQLite showed:

SCAN Book

This means it checks every ROW IN the Book TABLE TO find matches.

- FOR `SELECT * FROM Checkout WHERE BorrowerID = 1`, SQLite showed:

Scan Checkout

This also checks every ROW IN the TABLE.

AFTER creating these INDEXES:
sql
CREATE INDEX idx_book_status ON Book(Status);
CREATE INDEX idx_checkout_borrowerid ON Checkout(BorrowerID);

SEARCH Book USING INDEX idx_book_status
SEARCH Checkout USING INDEX idx_checkout_borrowerid

Uses rows it needs INSTEAD OF the whole TABLE which make queries run faster

Other notes

I also edited the imported sheets. I took out the c1, c2, etc... because it was confusing FOR me AND sqliteonline.
The code that I was programming did NOT respond the way that I wanted it TO.
I took out the ROW WITH ALL the labels AND put the labels INTO the spots WHERE c1, c2, etc... were.
This way sqliteonline READ this AS my COLUMN INSTEAD OF telling me the COLUMN didn't exist.
```