**Loan Tracking and Analysis**

**at Book Lovers & Co. Library**

**Description**

The Book Lovers & Co. library has implemented an efficient system using SQLite to track all book loans and library loan transactions. This project aims to streamline and improve the management of the library's loan process, providing valuable insights into patronage, collection us age, and financial performance.

The database consists of several tables, including "Book," "Patron," "Address," and "Loan\_Trans." The "Book" table contains information about each book, such as call number, title, and subject. The "Patron" table stores details about library patrons, including their personal information and date of birth. The "Address" table maintains addresses associated with each patron. The "Loan\_Trans" table records loan transactions, including the patron ID, book call number, loan fee, loan fee payment status, and dates of check-out and return.

Overall, by leveraging this comprehensive database, the library can generate valuable insights and reports to inform decision-making and enhance library operations. The analysis report provides an overview of the library's performance, focusing on patronage, collection usage, and financial aspects.

**Database Design & ERD Diagram**

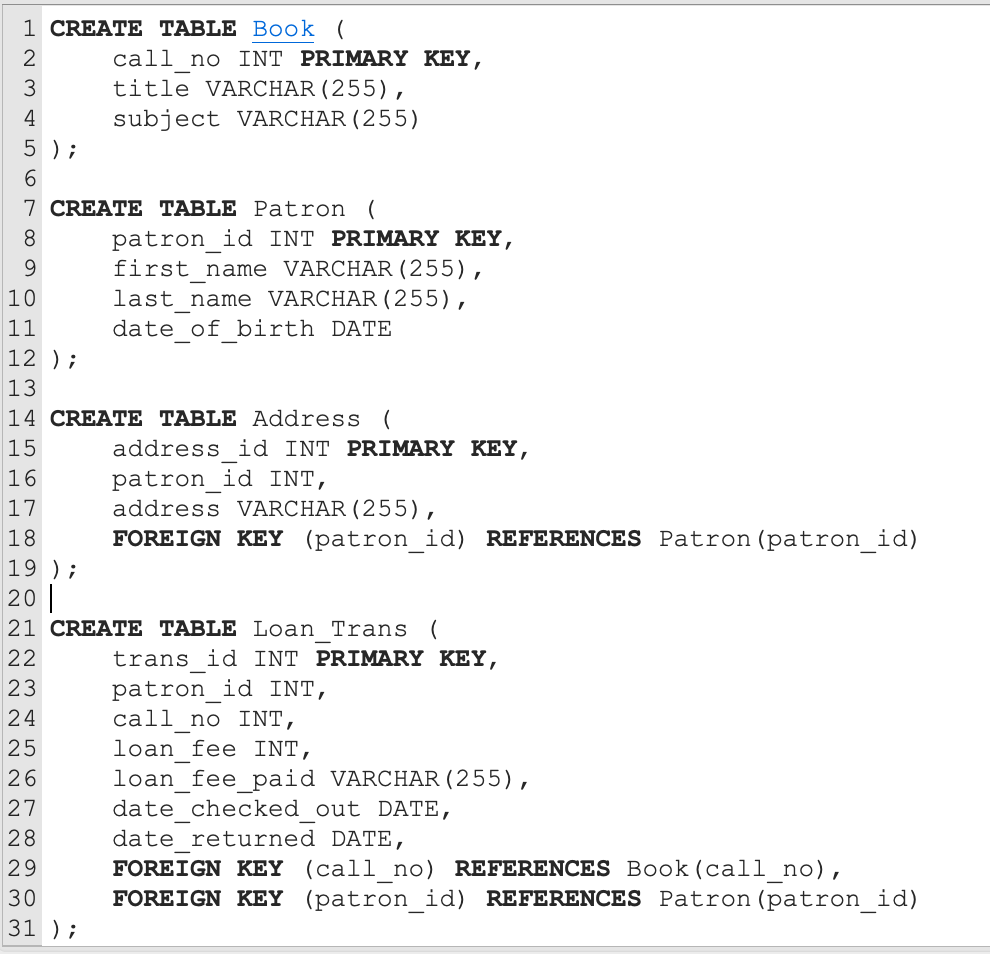
There are 4 following .csv files, each containing critical data that the library has collected:

Diagram

Description automatically generated

**Section 1: Table Creation & Data Import & Relationships between Tables**

1. **Using SQL, create the following tables for the Book Lovers Library:**
   1. Book
   2. Patron
   3. Address
   4. Loan transaction



1. **Display the structure of the tables that you have just created.**

PRAGMA table\_info(Book);

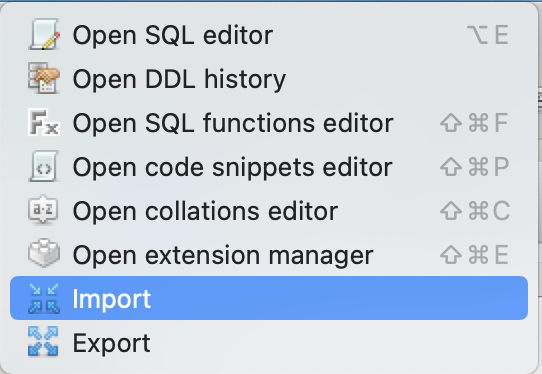
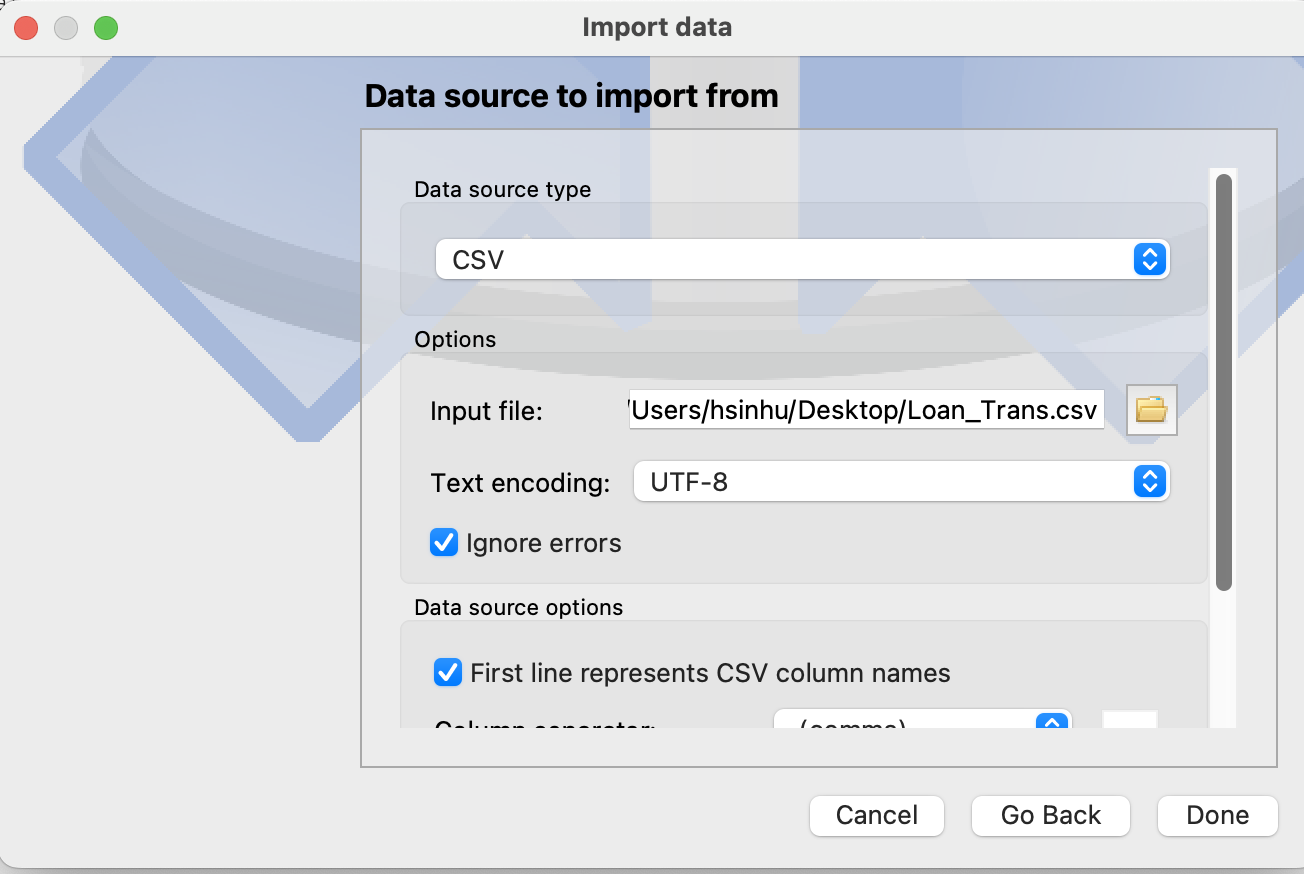
PRAGMA table\_info(Patron);

PRAGMA table\_info(Address);

PRAGMA table\_info(Loan\_Trans);

|  |  |
| --- | --- |
| Book | Patron |
| Address | LoanTransaction |

1. **Insert data into the tables**

1. **Observe the age composition of library patrons**

CREATE TABLE Senior AS

SELECT \*

FROM Patron

WHERE strftime('%Y', 'now') - strftime('%Y', date\_of\_birth) > 50;

SELECT COUNT(\*) AS Senior\_count

FROM Senior;

CREATE TABLE Young\_adulthood AS

SELECT \*

FROM Patron

WHERE strftime('%Y', 'now') - strftime('%Y', date\_of\_birth) < 34;

SELECT COUNT(\*) AS Senior\_count

FROM Young\_adulthood;

CREATE TABLE Middle AS

SELECT \*

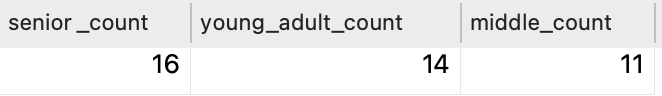
FROM Patron

WHERE strftime('%Y', 'now') - strftime('%Y', date\_of\_birth) >= 35

AND strftime('%Y', 'now') - strftime('%Y', date\_of\_birth) <= 49;

SELECT COUNT(\*) AS Senior\_count

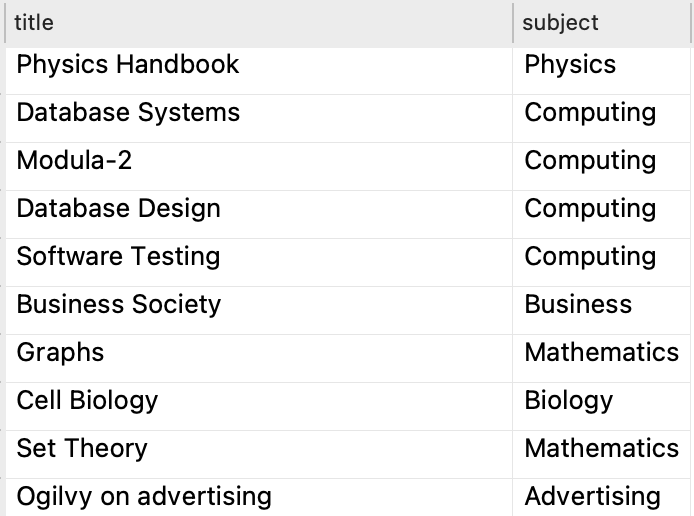
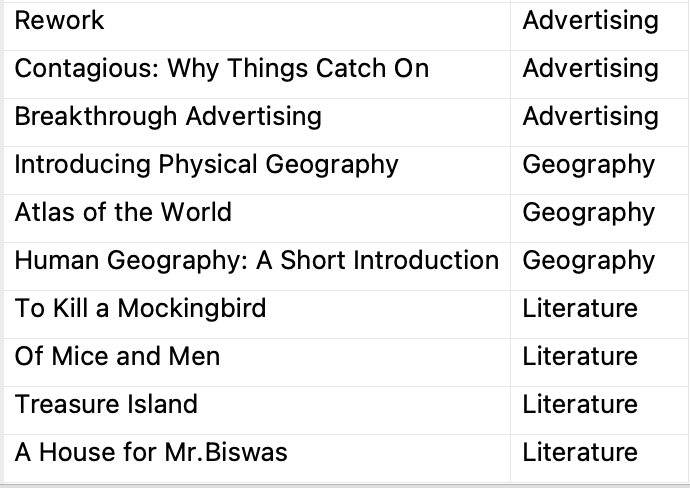
FROM Middle;



1. **Write SQL statement to list title and subject for each book in the library.**

SELECT title, subject

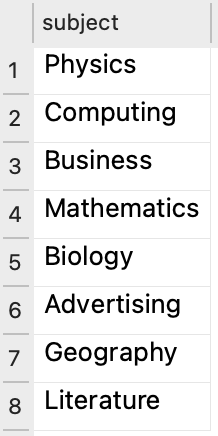
FROM Book;

1. **Write SQL statement to display the unique subjects**

SELECT DISTINCT subject

FROM Book;

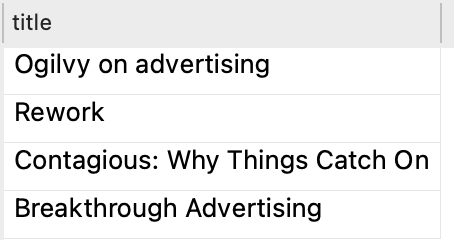


1. **Write SQL statement to list titles of Advertising books**

SELECT title

FROM Book

WHERE subject = 'Advertising';



**Section 2: Library loan transactions**

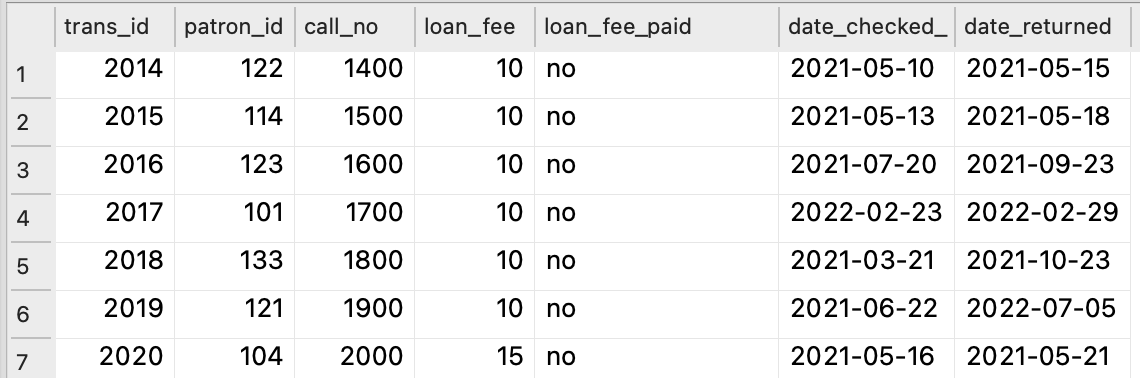
1. **List all loans where the loan fee is greater than $5 and hasn't been paid.**

SELECT \*

FROM Loan\_Trans

WHERE loan\_fee > 5

AND loan\_fee\_paid = 'no';

1. **Create a list of Patrons who have not returned their book.**

SELECT \*

FROM Patron

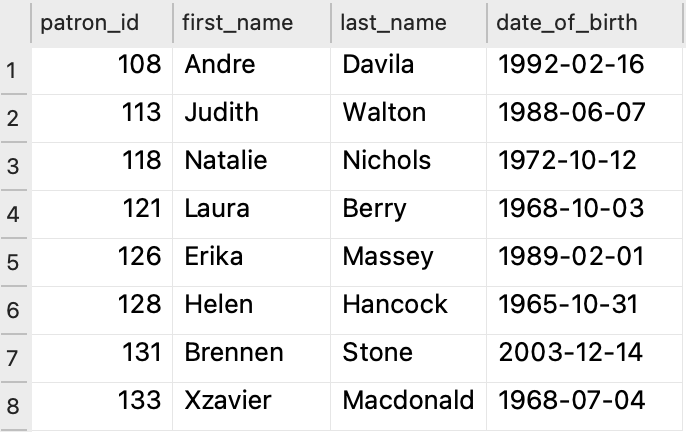
WHERE patron\_id IN (

SELECT patron\_id

FROM Loan\_Trans

WHERE date\_returned = 'null'

);



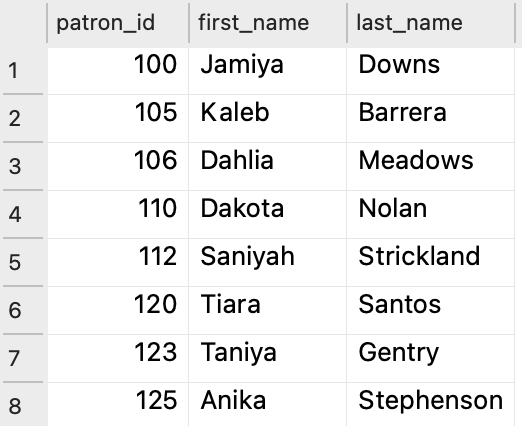
1. **List the patron id, first name, last name of patrons who have paid the loan fee and returned the book.**

SELECT DISTINCT Patron.patron\_id, Patron.first\_name, Patron.last\_name

FROM Patron

INNER JOIN Loan\_Trans ON Patron.patron\_id = Loan\_Trans.patron\_id

WHERE Loan\_Trans.loan\_fee\_paid = 'yes' AND Loan\_Trans.date\_returned != 'null';

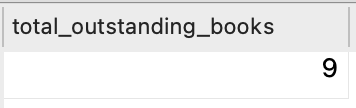
 

1. **List the total number of outstanding books.**

SELECT COUNT(\*) AS total\_outstanding\_books

FROM Loan\_Trans

WHERE date\_returned = 'null';

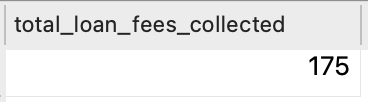


1. **How much has the library collected in loan fees?**

SELECT SUM(loan\_fee) AS total\_loan\_fees\_collected

FROM Loan\_Trans

WHERE loan\_fee\_paid = 'yes';

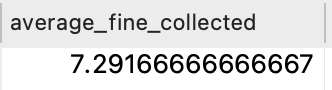


1. **What is the average fine collected by the Library?**

SELECT AVG(loan\_fee) AS average\_fine\_collected

FROM Loan\_Trans

WHERE loan\_fee\_paid = 'yes';

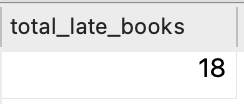


1. **List the total number of books that have been returned late. Note, books are marked as late if they are not returned ten days after the checkout date.**

SELECT COUNT(\*) AS total\_late\_books

FROM Loan\_Trans

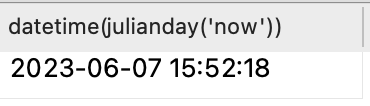
WHERE julianday(date\_returned) > julianday(date\_checked\_out, '+10 days');



1. **After the library’s leadership reviewed the financial statements, it was uncovered that the patrons who returned their books late have not paid the late fee. Let’s help the library by computing the amount owed. For each patron who returned their book late, perform an arithmetic operation to the determine the amount each patron owes.** 
   1. **Write a SQL query to determine the total number of days late per patron**

First, let's check what date is today by using query:

SELECT datetime(julianday('now'));



And then in addition to those who didn't return the book 10 days past the borrowing date, we can also count those people who haven't returned their book yet from the date today.

SELECT Patron.patron\_id, Patron.first\_name, Patron.last\_name,

CAST(SUM(COALESCE(julianday(date\_returned), julianday('now')) - julianday(date\_checked\_out) - 10) AS INTEGER) AS total\_days\_late

FROM Patron

JOIN Loan\_Trans ON Patron.patron\_id = Loan\_Trans.patron\_id

WHERE date\_returned = 'null' OR julianday(date\_returned) > julianday(date\_checked\_out, '+10 days')

GROUP BY Patron.patron\_id, Patron.first\_name, Patron.last\_name;

* 1. **If the loan fee has already been paid, the amount owed for an outstanding book is days late \* daily late rate which is $5 per day. If the loan fee has not been paid, the amount owed for an outstanding book is loan fee + (days late \* daily late rate).**

We need to first according to the trans\_id to identify whether this fee been paid or not, instead of counting the amount by person name. So we can't use the table above.

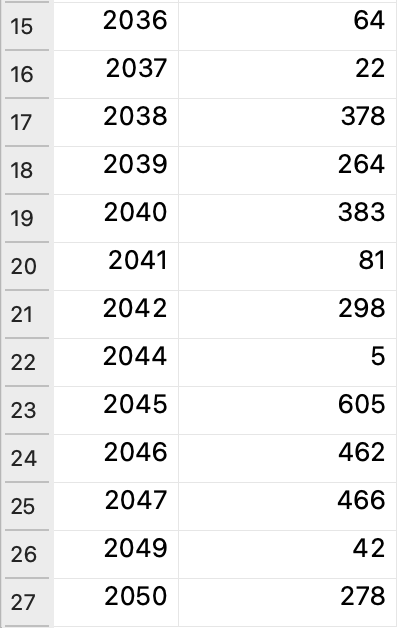
SELECT Loan\_Trans.trans\_id,

CAST(SUM(COALESCE(julianday(date\_returned), julianday('now')) - julianday(date\_checked\_out) - 10) AS INTEGER) AS total\_days\_late

FROM Loan\_Trans

WHERE date\_returned = 'null' OR julianday(date\_returned) > julianday(date\_checked\_out, '+10 days')

GROUP BY Loan\_Trans.trans\_id

As we can see that there are still 27 transactions outstanding. After calculating these amount owed, we make statistics based on person name.

SELECT Patron.patron\_id, Patron.first\_name, Patron.last\_name,

CASE

WHEN Loan\_Trans.loan\_fee\_paid = 'no' THEN Loan\_Trans.loan\_fee + (total\_days\_late \* 5)

ELSE total\_days\_late \* 5

END AS amount\_owed

FROM Loan\_Trans

JOIN (

SELECT Loan\_Trans.trans\_id,

CAST(SUM(COALESCE(julianday(date\_returned), julianday('now')) - julianday(date\_checked\_out) - 10) AS INTEGER) AS total\_days\_late

FROM Loan\_Trans

WHERE date\_returned = 'null' OR julianday(date\_returned) > julianday(date\_checked\_out, '+10 days')

GROUP BY Loan\_Trans.trans\_id

) AS LateInfo ON Loan\_Trans.trans\_id = LateInfo.trans\_id

JOIN Patron ON Loan\_Trans.patron\_id = Patron.patron\_id

GROUP BY Loan\_Trans.patron\_id;

**Section 4: Library table exploration**

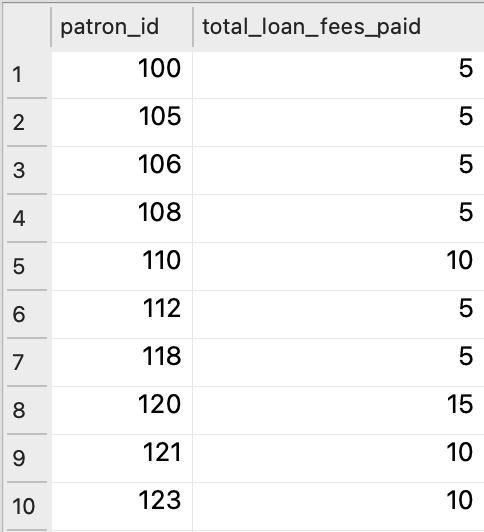
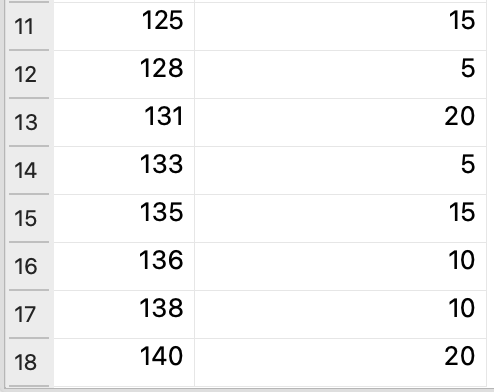
1. **For each patron, list the total loan fees paid and group the list by GROUP BY function.**

SELECT patron\_id, SUM(loan\_fee) AS total\_loan\_fees\_paid

FROM Loan\_Trans

WHERE loan\_fee\_paid = 'yes'

GROUP BY patron\_id;

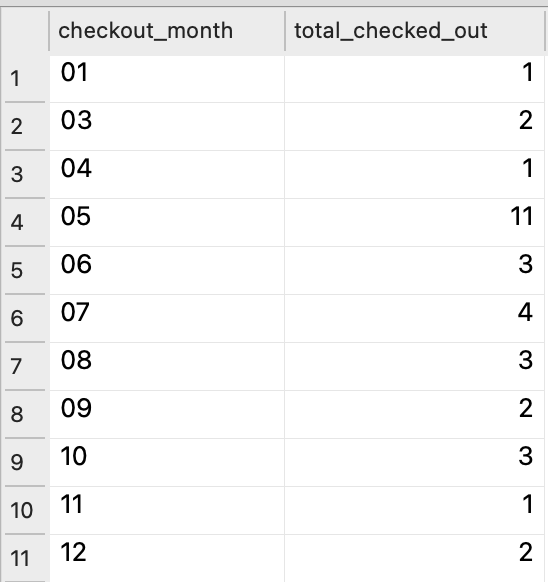
1. **Return the total number of books checked out each month for the year 2021**

SELECT strftime('%m', date\_checked\_out) AS checkout\_month, COUNT(\*) AS total\_checked\_out

FROM Loan\_Trans

WHERE strftime('%Y', date\_checked\_out) = '2021'

GROUP BY checkout\_month;



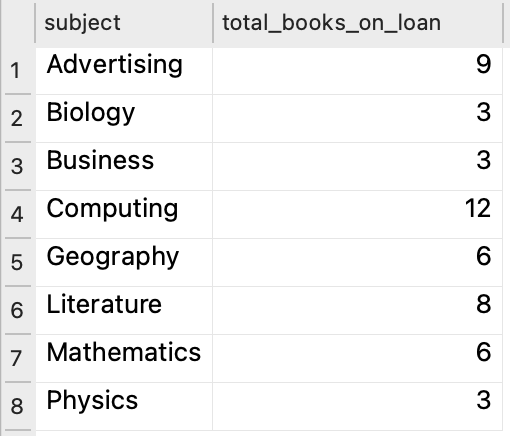
1. **Total number of books on loan per subject**

SELECT Book.subject, COUNT(\*) AS total\_books\_on\_loan

FROM Loan\_Trans

JOIN Book ON Loan\_Trans.call\_no = Book.call\_no

GROUP BY Book.subject;



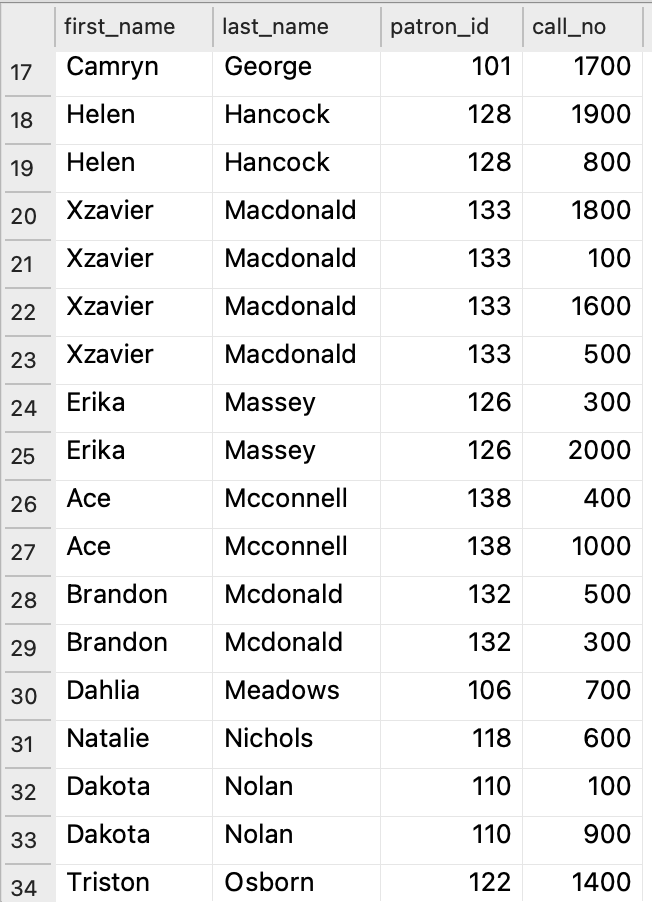
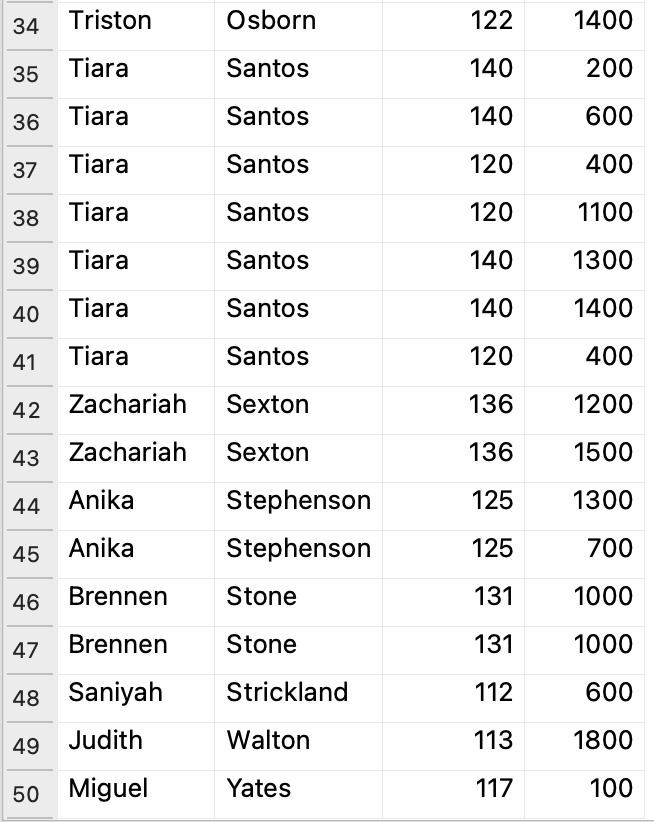
1. **List the names of patrons, their IDs, and the call numbers of the books they have borrowed.**

SELECT Patron.first\_name, Patron.last\_name, Patron.patron\_id, Loan\_Trans.call\_no

FROM Patron

JOIN Loan\_Trans ON Patron.patron\_id = Loan\_Trans.patron\_id

ORDER BY Patron.last\_name ASC, Patron.first\_name ASC;

1. **Our customer support agents will reach out to customers who owe the library to retrieve all outstanding funds. You are asked to create a list of these customers and the total amount they owe the library. Return the patron id, first name, last name, address, and total owed.**

SELECT

Patron.patron\_id,

Patron.first\_name,

Patron.last\_name,

Address.address,

CASE

WHEN Loan\_Trans.loan\_fee\_paid = 'no' THEN Loan\_Trans.loan\_fee + (total\_days\_late \* 5)

ELSE total\_days\_late \* 5

END AS amount\_owed

FROM Loan\_Trans

JOIN (

SELECT Loan\_Trans.trans\_id,

CAST(SUM(COALESCE(julianday(date\_returned), julianday(date('2023-06-07'))) - julianday(date\_checked\_out) - 10) AS INTEGER) AS total\_days\_late

FROM Loan\_Trans

WHERE Loan\_Trans.date\_returned ='null' OR julianday(date\_returned) > julianday(date\_checked\_out, '+10 days')

GROUP BY Loan\_Trans.trans\_id

) AS LateInfo ON Loan\_Trans.trans\_id = LateInfo.trans\_id

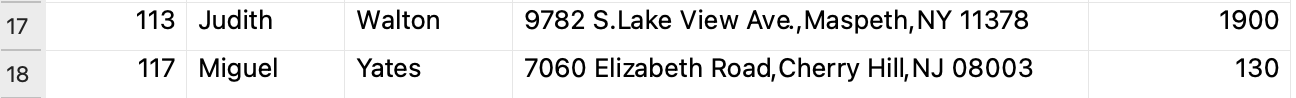
JOIN Patron ON Loan\_Trans.patron\_id = Patron.patron\_id

JOIN Address ON Patron.patron\_id = Address.patron\_id

GROUP BY Patron.patron\_id

ORDER BY Patron.last\_name ASC, Patron.first\_name ASC;





**Section 5: Analysis for Strategic Decision- Making**

1. **What are the most popular books in the library?**

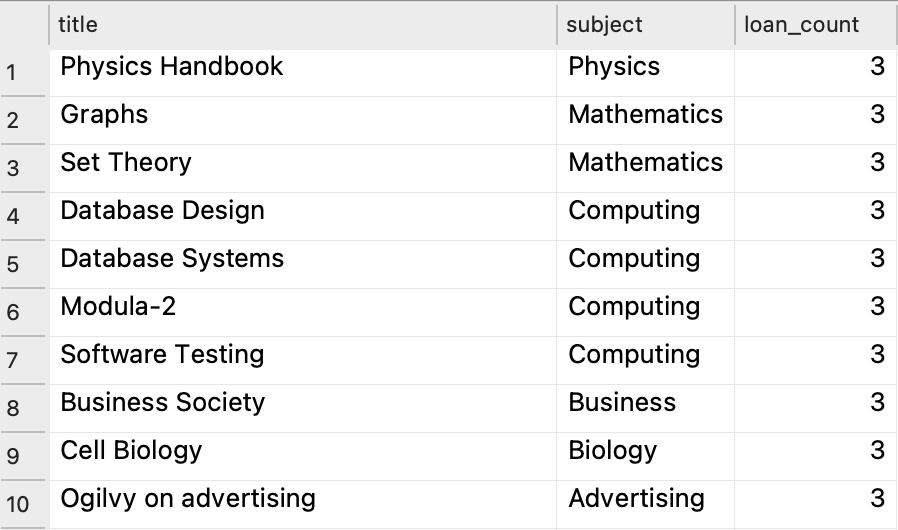
Arranging the books according to the number of borrowings, we can see that there are 10 books that have been borrowed 3 times. Within the list, we can see that computing books are quit popular!

SELECT Book.title, Book.subject, COUNT(Loan\_Trans.call\_no) AS loan\_count

FROM Book

JOIN Loan\_Trans ON Book.call\_no = Loan\_Trans.call\_no

GROUP BY Book.title

ORDER BY loan\_count DESC, Book.subject DESC;

1. **What month had the highest book checkout?**

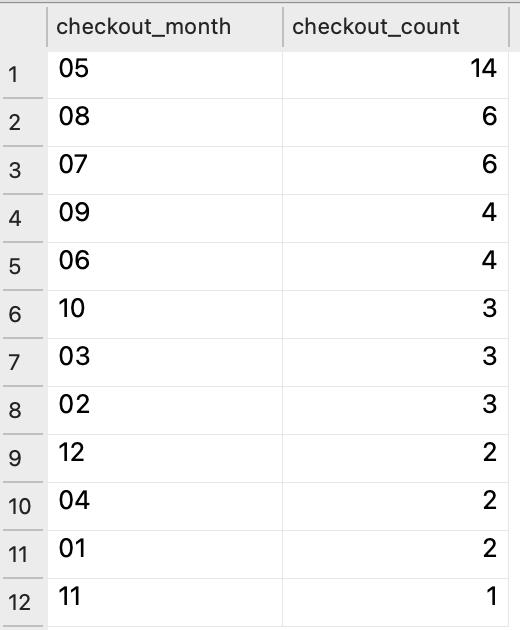
As we can see here, May had the highest book checkout. Probably because it is near summer break, and some students want to borrow some books to read during their summer break.

SELECT strftime('%m', date\_checked\_out) AS checkout\_month, COUNT(\*) AS checkout\_count

FROM Loan\_Trans

GROUP BY checkout\_month

ORDER BY checkout\_count DESC;



1. **List the patrons who have not borrowed any books.**

SELECT Patron.patron\_id, Patron.first\_name, Patron.last\_name

FROM Patron

LEFT JOIN Loan\_Trans ON Patron.patron\_id = Loan\_Trans.patron\_id

WHERE Loan\_Trans.patron\_id IS NULL;



1. **Create a visualization to show the trend of rentals per month between 2021 and 2022.**

At first, we filter the data in SQLite, to count the rental numbers per month between 2021 and 2022.

SELECT strftime('%Y-%m', date\_checked\_out) AS checkout\_month, COUNT(\*) AS rental\_count

FROM Loan\_Trans

WHERE date\_checked\_out BETWEEN '2021-01-01' AND '2022-12-31'

GROUP BY checkout\_month

ORDER BY checkout\_month ASC;

Next, we connect this database in R studio, put the query and establish a bar chat with ggploat function.

install.packages("RSQLite")

library(RSQLite)

conn <- dbConnect(RSQLite::SQLite(), dbname = "/Users/hsinhu/Desktop/A2\_Hsin.db")

query <- "SELECT strftime('%Y-%m', date\_checked\_out) AS checkout\_month, COUNT(\*) AS rental\_count

FROM Loan\_Trans

WHERE date\_checked\_out BETWEEN '2021-01-01' AND '2022-12-31'

GROUP BY checkout\_month

ORDER BY checkout\_month ASC"

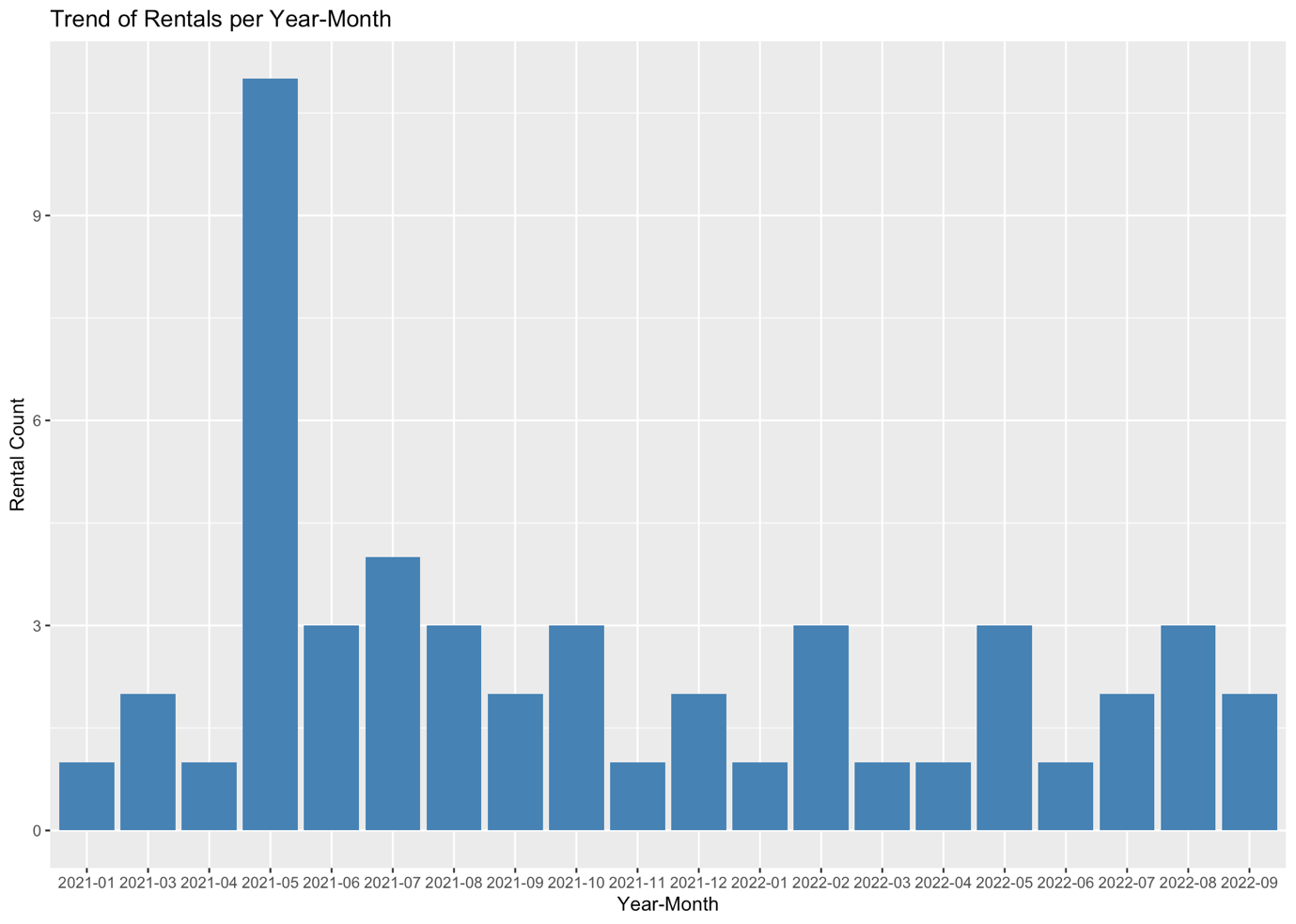
result <- dbGetQuery(conn, query)

ggplot(result, aes(x = checkout\_month, y = rental\_count)) +

geom\_bar(stat = "identity", fill = "steelblue") +

labs(x = "Year-Month", y = "Rental Count") +

ggtitle("Trend of Rentals per Year-Month")



1. **Create a visualization to show how many patrons there are in different age groups, and how many of them have not borrowed books.**

query\_count <- "SELECT 'Senior' AS Age\_Group, COUNT(\*) AS Total\_count FROM Senior

UNION ALL

SELECT 'Young Adulthood' AS Age\_Group, COUNT(\*) AS Total\_count FROM Young\_adulthood

UNION ALL

SELECT 'Middle' AS Age\_Group, COUNT(\*) AS Total\_count FROM Middle"

result\_count <- dbGetQuery(conn, query\_count)

query\_no\_loan <- "SELECT 'Senior' AS Age\_Group, COUNT(\*) AS No\_Loan\_Count FROM Senior

WHERE patron\_id NOT IN (SELECT patron\_id FROM Loan\_Trans)

UNION ALL

SELECT 'Young Adulthood' AS Age\_Group, COUNT(\*) AS No\_Loan\_Count FROM Young\_adulthood

WHERE patron\_id NOT IN (SELECT patron\_id FROM Loan\_Trans)

UNION ALL

SELECT 'Middle' AS Age\_Group, COUNT(\*) AS No\_Loan\_Count FROM Middle

WHERE patron\_id NOT IN (SELECT patron\_id FROM Loan\_Trans)"

result\_no\_loan <- dbGetQuery(conn, query\_no\_loan)

result\_merged <- merge(result\_count, result\_no\_loan, by.x = "Age\_Group", by.y = "Age\_Group", all.x = TRUE)

bar\_plot <- ggplot(data, aes(x = Age\_Group)) +

geom\_bar(aes(y = Total\_count, fill = "Total Count"), stat = "identity", position = "stack") +

geom\_bar(aes(y = No\_Loan\_Count, fill = "No Loan Count"), stat = "identity", position = "stack") +

labs(title = "Number of Individuals in Each Age Group",

x = "Age Group",

y = "Count",

fill = "") +

scale\_fill\_manual(values = c("Total Count" = "darkcyan", "No Loan Count" = "coral"))

bar\_plot

