**18.5.2. Warm-up Queries**

Write the following queries to get warmed up.

1. **Return the mystery book titles and their ISBNs.**

select \* from genre inner join book on genre.genre\_id = book.genre\_id where genres = "Mystery";

SELECT title, isbn

FROM book

WHERE genre\_id IN (SELECT genre\_id FROM genre WHERE genres like "Mystery");

Graphical user interface, application, email

Description automatically generated

1. **Return all of the titles and author’s first and last names for books written by authors who are currently living.**

select title, last\_name, first\_name from author inner join book on author.author\_id = book.author\_id where deathday is null;

Table

Description automatically generated

**18.5.3. Loan Out a Book**

**A big function that you need to implement for the library is a script that updates the database when a book is loaned out.**

**The script needs to perform the following functions:**

1. **Change available to FALSE for the appropriate book.**
2. Add a new row to the loan table with today’s date as the date\_out and the ids in the row matching the appropriate patron\_id and book\_id.

INSERT INTO loan(patron\_id, date\_out, book\_id)

VALUE (1,now(),1), (3,curdate(),5),(4,now(),7), (9,curdate(),9);

Table

Description automatically generated

1. Update the appropriate patron with the loan\_id for the new row created in the loan table.

update patron set loan\_id = 2 where patron\_id = 3;

update patron set loan\_id = 1 where patron\_id = 1;

or

UPDATE patron

SET loan\_id = (SELECT loan\_id

FROM loan

WHERE loan.patron\_id = patron.patron\_id)where first\_name = “gds” and last name = “jelk”;

Or

UPDATE patron

SET loan\_id = (SELECT loan\_id

FROM loan

WHERE

loan.patron\_id = patron.patron\_id)

where

patron\_id = (select patron\_id from loan where book\_id = (select book\_id from book where title = "Beloved"));

Table

Description automatically generated

You can use any patron and book that strikes your fancy to create this script!

**18.5.4. Check a Book Back In**

Working with the same patron and book, create the new script!

The other key function that we need to implement is checking a book back in. To do so, the script needs to:

1. Change available to TRUE for the appropriate book.

update book

set available = true where book\_id = 34 ;

Graphical user interface, text, application

Description automatically generated

1. Update the appropriate row in the loan table with today’s date as the date\_in.

select @lid:= (select loan\_id from loan where book\_id = (select book\_id from book where title = "Beloved"));

update loan

set date\_in = curdate()

where

loan\_id = @lid;

or

select @pid:= (select patron\_id from loan where book\_id = (select book\_id from book where title = "Beloved"));

update loan

set date\_in = curdate()

where

loan\_id = @pid;



1. Update the appropriate patron changing loan\_id back to null.

UPDATE patron

SET loan\_id = null

where

patron\_id = (select patron\_id from loan where book\_id = (select book\_id from book where title = "Beloved"));

Graphical user interface, application

Description automatically generated

Once you have created these scripts, loan out 5 new books to 5 different patrons.

**18.5.5. Wrap-up Query**

Select first\_name , last\_name, genres

From (select patron.first-name, patron.last\_name, loan.book\_id

From patron

Inner join loan

On loan.patron\_id = patron.patron\_id

)

Inner join (select \* from book

Inner join genre

On book.genre\_id = genre.genre\_id

) as book\_genre

On patron\_loan.book\_id = book\_genre.bookid

Or

SELECT patron\_loan.first\_name, patron\_loan.last\_name, genre\_book.genres

FROM (

SELECT patron.first\_name, patron.last\_name, loan.book\_id

FROM patron

INNER JOIN loan ON loan.loan\_id = patron.loan\_id

) AS patron\_loan

INNER JOIN (

SELECT genre.genre\_id, genre.genres, book.book\_id

FROM genre

INNER JOIN book ON genre.genre\_id = book.genre\_id) AS genre\_book

ON genre\_book.book\_id = patron\_loan.book\_id;

**18.5.6. Bonus Mission**

1. Return the counts of the books of each genre. Check out the [documentation](https://dev.mysql.com/doc/refman/8.0/en/counting-rows.html) to see how this could be done!

select genre\_id , title , count(\*) as book\_count from book group by genre\_id;

Table

Description automatically generated

2 . A reference book cannot leave the library. How would you modify either the reference\_book table or the book table to make sure that doesn’t happen? Try to apply your modifications.

select @bid:= (select book\_id from reference\_books);

UPDATE book

SET available = true

WHERE book\_id = @bid;

Or

select @bid:= (select book\_id from reference\_books);

update book

set available = case

when genre\_id = 25 then true

else

false

end

WHERE book\_id = @bid;

Table

Description automatically generated