CSC 4710/6710 FINAL PROJECT

SPRING 2021

BRYANNA HARDY

Contents

PHASE 1

I chose to create a Library Management Database, because I enjoy going into antique bookstores, libraries, or any bookstores, just to look at them, read some in stores, if allowed, and to buy. This database can be used to store important information for a library and help keep the data organized. Here is the start to my Library Management Database down below:

Local Town Library is a local library that is starting to expand rapidly, and its owner has finally decided it was time to store information about its readers, books, and its data as well in a database. This database will help us store the reader's information, book titles and their categories, library branches, and much more. Here are some of Local Town Library's requirements:

Each employee at Local Town Library has an employee id, name, and branch id. They keep track of the books in the library, and they can only work at one branch.

Each branch has a branch id, manager id, and a city location.

Each reader has a name, customer id, that is unique to them and cannot be null, the date when they first registered, and how many books are issued to them currently.

Every time a book is checked out, we store an issue date, where it stores the book name, customer id, ISBN, the date it was issued, the due date, and issue id.

Each book has a title, ISBN, which is unique, genre id, and a shelf number.

Each category is issued with a genre such as, comedy, action, suspense, fiction, non-fiction, and fantasy, a genre id, which is unique, and a shelf number.

Each shelf is issued a shelf number, which is unique, and floor number.

Each author has a name and a publisher.

The readers can borrow up to 4 books maximum if they would like, it is not required. They also can visit the other branches.

The books are located on shelves by categories and have many categories. Books can also be located at multiple branches.

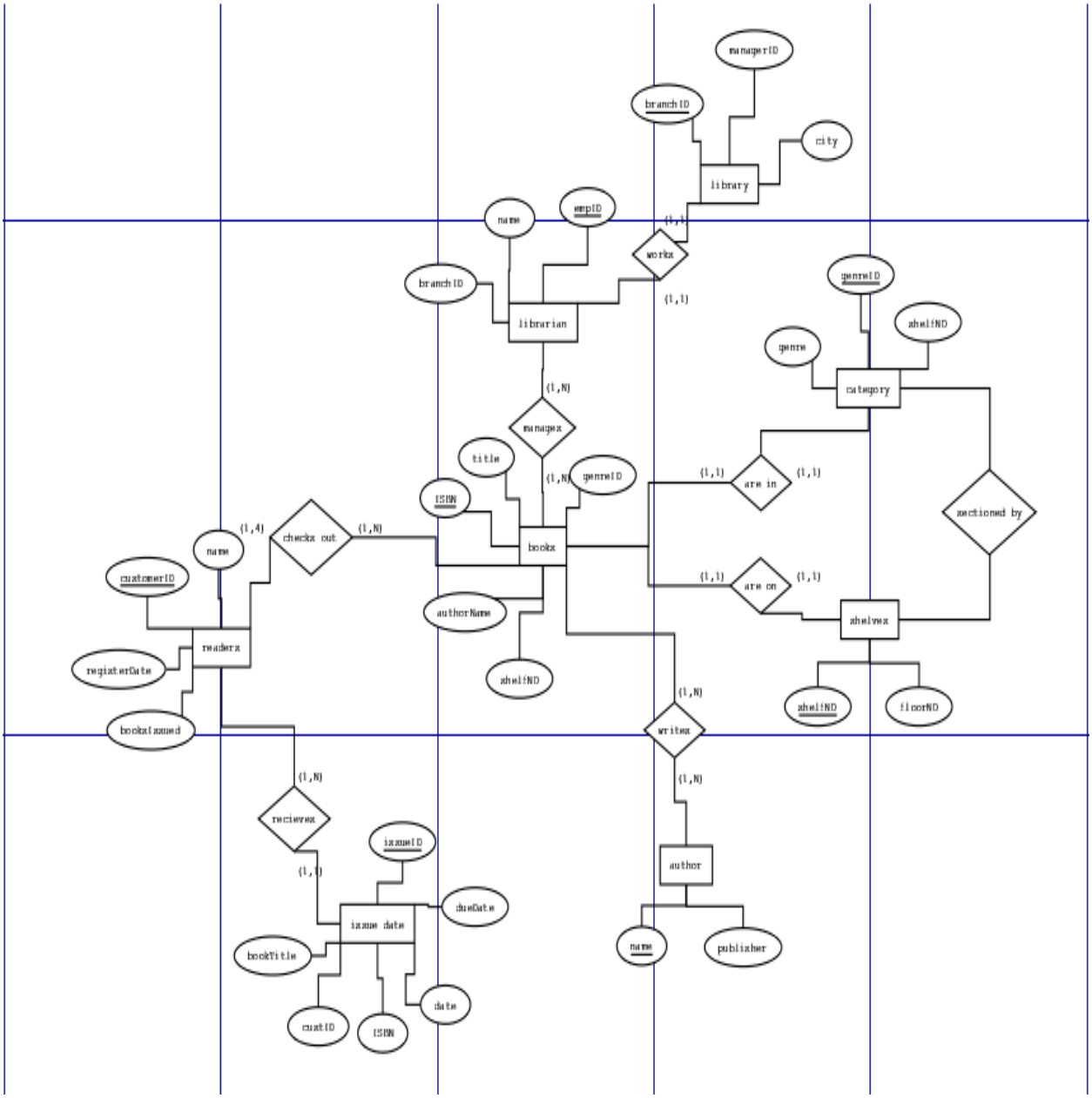
Authors can write more than one book.

Functional Requirements:

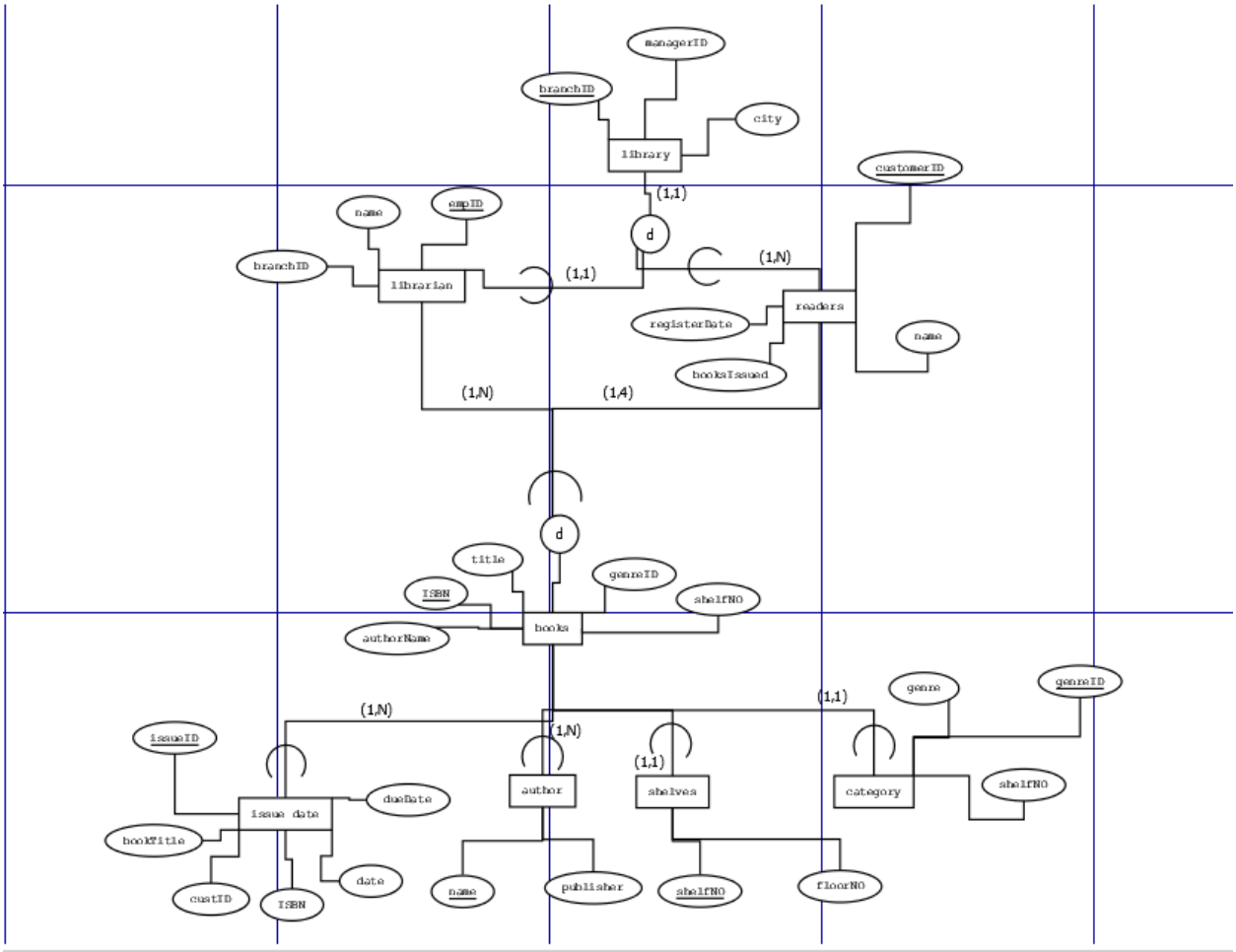
- Need to list of the people who due date is on a [certain date] along with the book name.
- Need to list the names of the people who is currently checking out a certain book.
- Need to list the author's book titles, name, and publishers AND create a view.
- Need to list each category/genre with the name of book and author AND a view.
- Need to list the name of the librarians that work at a certain location.
- Need to create a trigger to check if the register date is greater than the current date, if so, it is invalid.
- Need to create a trigger to check for null publishers, if it has not been put into the system, if so, send a message to update system.

PHASE 2

ER Diagram:

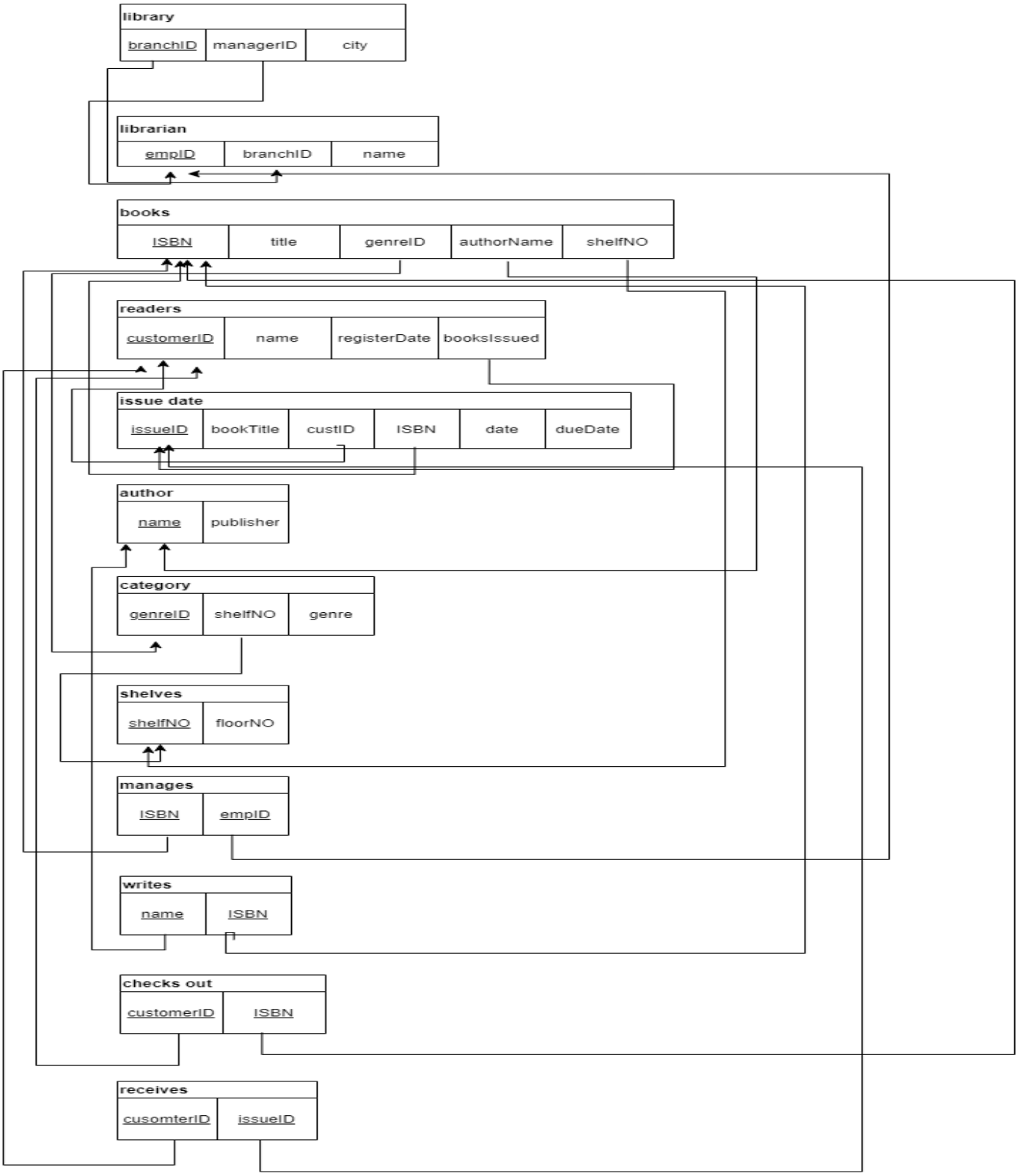


EER Diagram:


managerID
branchID
city
library
customerID
(1,1)
name
empID
d
branchID
librarian
(1,1)
(1,N)
registerDate
readers
name
booksIssued
(1,N)
(1,4)
d
title
genreID
ISBN
shelfNO
authorName
books
genre
genreID
(1,N)
(1,1)
(1,N)
issueID
(1,1)
author
shelves
category
shelfNO
issue date
dueDate
bookTitle
date
custID
ISBN
name
publisher
shelfNO
floorNO

PHASE 3

Relational Schema:



For each entity, I created their very own table, along with the attributes that is associated with them. In addition, for each foreign key, I created another table to represent what they derived from and map those to its respectful primary key from the parent entity/key. For example, the relationships with a many to many relationships, they receive their own table with the attributes.

PHASE 4

Data Dictionary:

Table	*Attribute*	*Data Type*	Primary Key	Foreign Key	Constraints
LIBRARY	branchID	INT	YES		NOT NULL
LIBRARY	managerID	INT		LIBRARIAN(empID)	
LIBRARY	city	VARCHAR(25)			
LIBRARIAN	empID	INT	YES		NOT NULL
LIBRARIAN	name	VARCHAR(100)			
LIBRARIAN	branchID	INT		LIBRARY(branchID)	
BOOKS	ISBN	INT	YES		NOT NULL
BOOKS	title	VARCHAR(100)			
BOOKS	genreID	INT		CATEGORY(genreID)	
BOOKS	authorName	VARCHAR(100)			
BOOKS	shelfNO	INT		SHELVES(shelfNO)	
CATEGORY	genreID	INT	YES		
CATEGORY	genre	VARCHAR(50)			
CATEGORY	shelfNO	INT		SHELVES(shelfNO)	
SHELVES	shelfNO	INT	YES		NOT NULL
SHELVES	floorNO	INT			
AUTHOR	name	VARCHAR(100)	YES		NOT NULL
AUTHOR	publisher	VARCHAR(150)			
READERS	customerID	INT	YES		NOT NULL
READERS	name	VARCHAR(100)			
READERS	registerDate	DATE			<= CURRENT DATE
READERS	booksIssued	INT		ISSUEDATE(issueID)	
ISSUE DATE	issueID	INT	YES		NOT NULL
ISSUE DATE	dueDate	DATE			
ISSUE DATE	book	VARCHAR(100)			
ISSUE DATE	custID	INT		READERS(customerID)	
ISSUE DATE	ISBN	INT		BOOKS(ISBN)	
ISSUE DATE	currDate	DATE			

PHASE 5

Implementation:

****ALTER TABLE IS NOT RAN UNTIL ALL CREATE TABLES AND INSERT VALUES HAS BEEN INSERTED INTO DATABASE****

```
CREATE TABLE LIBRARY (
   BRANCHID     INT              NOT NULL,
   MANAGERID  INT,
   CITY            VARCHAR(25),
   PRIMARY KEY (BRANCHID)
);
ALTER TABLE LIBRARY
ADD CONSTRAINT fk_manager
FOREIGN KEY (MANAGERID) REFERENCES LIBRARIAN(EMPID);


CREATE TABLE LIBRARIAN (
   EMPID                        INT                          NOT NULL,
   NAME                         VARCHAR(100),
   BRANCHID              INT,
   PRIMARY KEY (EMPID)
);
ALTER TABLE  librarian
ADD CONSTRAINT fk_branchID
FOREIGN KEY (BRANCHID) REFERENCES LIBRARY(BRANCHID);
```

```
CREATE TABLE BOOKS (
   ISBN              INT                          NOT NULL,
   TITLE             VARCHAR(100),
   GENREID           INT,
   AUTHORNAME              VARCHAR(100),
   SHELFNO           INT,
   PRIMARY KEY (ISBN)
);
ALTER TABLE  books
ADD CONSTRAINT fk_books
FOREIGN KEY (GENREID) REFERENCES CATEGORY(GENREID),
ADD FOREIGN KEY (SHELFNO) REFERENCES SHELVES(SHELFNO);

CREATE TABLE CATEGORY (
   GENREID           INT                          NOT NULL,
   GENRE                   VARCHAR(50),
   SHELFNO           INT,
   PRIMARY KEY (GENREID)
);
ALTER TABLE  category
ADD CONSTRAINT fk_category
FOREIGN KEY (SHELFNO) REFERENCES SHELVES(SHELFNO);
```

```
CREATE TABLE SHELVES (
   SHELFNO      INT              NOT NULL,
   FLOORNO      INT,
   PRIMARY KEY (SHELFNO)
);

CREATE TABLE AUTHOR (
   NAME                 VARCHAR(100)          NOT NULL,
   PUBLISHER    VARCHAR(150),
   PRIMARY KEY (NAME)
);

CREATE TABLE READERS (
   CUSTOMERID                INT                              NOT NULL,
   NAME                           VARCHAR(100),
   REGISTERDATE         DATE,
   BOOKSISSUED          INT,
   PRIMARY KEY (CUSTOMERID)
);
ALTER TABLE  readers
ADD CONSTRAINT fk_readers
FOREIGN KEY (BOOKSISSUED) REFERENCES ISSUEDATE(ISSUEID);
```

```
CREATE TABLE ISSUEDATE (
   ISSUEID        INT                              NOT NULL,
   DUEDATE       DATE,
   BOOK                  VARCHAR(100),
   CUSTID                INT,
   ISBN            INT,
   CURRDATE    DATE,
   PRIMARY KEY (ISSUEID)
);


ALTER TABLE  issuedate
ADD CONSTRAINT fk_issue
FOREIGN KEY (CUSTID) REFERENCES READERS(CUSTOMERID),
ADD FOREIGN KEY (ISBN) REFERENCES BOOKS(ISBN);
```

****ALTER TABLE IS NOT RAN UNTIL ALL CREATE TABLES AND INSERT VALUES HAS BEEN INSERTED INTO DATABASE****

SQL QUERIES:

#list all of the names of the people who is currently checking out the book the Magicians.

```
select readers.NAME
from (readers join issuedate on readers.CUSTOMERID = issuedate.CUSTID)
where issuedate.BOOK = 'The Magicians';
```

#list all of the people who due date is on may 10 of 2021 and the book name.

```
select readers.NAME, issuedate.BOOK
from (readers join issuedate on readers.CUSTOMERID = issuedate.CUSTID)
where issuedate.DUEDATE = '2021-05-10';
```

#list the author's book title, name, and publishers

```
select distinct books.TITLE, author.NAME, author.PUBLISHER
from (author join books on author.name = books.AUTHORNAME);
```

#list each category/genre with the name of book and author

```
select distinct category.GENRE, books.TITLE, books.AUTHORNAME
from (category join books on category.GENREID = books.GENREID);
```

#list the name of the librarians that work at ashyville

```
select librarian.NAME
from (librarian join library on librarian.BRANCHID = library.BRANCHID)
where library.CITY = 'ASHYVILLE';
```

PHASE 6

Implementation:

Views:

#list the author's book title, name, and publishers

```
create view authorBooks
as select
        B.title,
    A.name,
    A.publisher
from
        author as A
inner join
        books as B on A.name = B.authorname;

select * from authorBooks;
```

##shows the results

#list each category/genre with the name of book and author

```
create view categories
as select
        C.genre,
    B.title,
    B.authorname
from
        category as C
inner join
        books as B on C.genreid = B.genreid;
select * from categories;
```

shows the results

Triggers:

Needs to check if the register date is greater than the current date, if so, it is invalid.

```
DROP TRIGGER IF EXISTS `librarydb`.`readers_BEFORE_INSERT`;


DELIMITER $$

USE `librarydb`$$

CREATE DEFINER = CURRENT_USER TRIGGER `librarydb`.`readers_BEFORE_INSERT` BEFORE INSERT ON `readers` FOR EACH ROW

BEGIN

declare msg varchar(255);

if new.registerdate > curdate() then

set msg = 'INVALID DATE';

signal sqlstate '45000' set message_text = msg;

end if;

END$$

DELIMITER ;
```

INSERT INTO READERS VALUES ('154', 'Mark East', '2025-01-12', '521'); ## trigger

Need to create a trigger to check for null publishers, if it hasn't been put into the system, if so, send a message to update system.

```
DROP TRIGGER IF EXISTS `librarydb`.`author_AFTER_INSERT`;


DELIMITER $$

USE `librarydb`$$

CREATE DEFINER=`root`@`localhost` TRIGGER `author_AFTER_INSERT` AFTER INSERT ON
`author` FOR EACH ROW

BEGIN

declare msg varchar(255);

if new.publisher is null then

set msg = 'Please, set a publisher.';

signal sqlstate '45000' set message_text = msg;

end if;

END$$

DELIMITER ;
```

insert into author values ('Chris Lee', null); #sets trigger

SUMMARY

For the final project, I have created a Library Management Database System. This Library Management System stores all important information that is important, such as the employee's information, book information, authors, publishers, and much more. For example, it should output the people who due date is on a [certain date] along with the book name. It also should create a view that outputs all the categories and genres along with the author's name and title of the book. It should also be capable of creating a trigger that outputs an error message if a publisher is set to null.