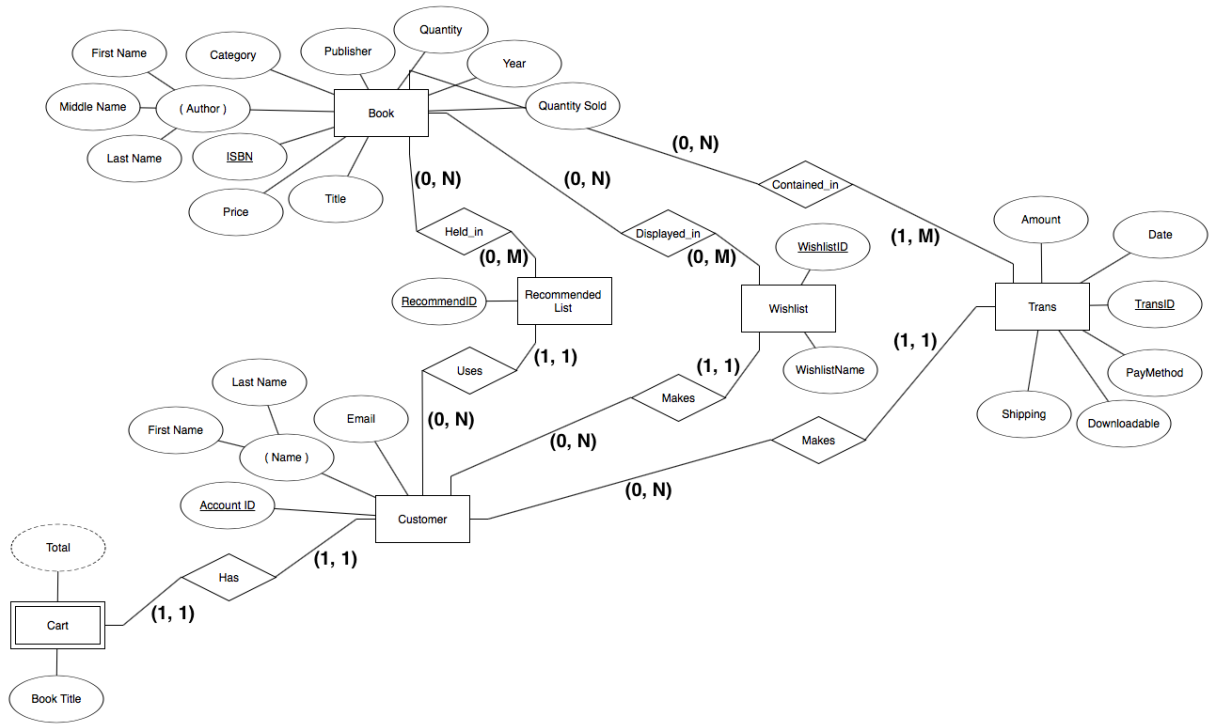
Section 1 - Database Description

Our database for the bookstore is designed around the relationships between the various aspects of a bookstore: the books themselves, customers, transactions, and other features.

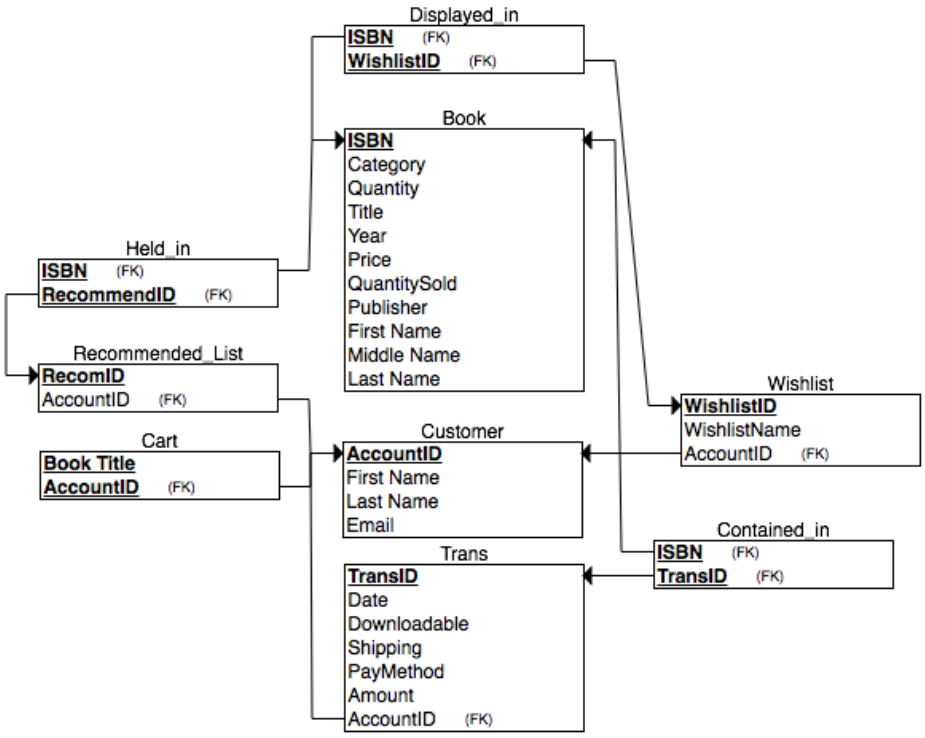
a) Below is the final version of our Entity Relationship (ER) Diagram that has been updated and edited throughout the semester.



Descriptions of Key Entities

Book:	A book. All books have a unique ISBN, title, and author's last name who wrote the book. Books have additional attributes, such as year and quantity, that are not necessary in order for a book to be entered into the database.
Cart:	An online shopping cart. Each cart needs a title of book within the cart and ID.
Customer:	A customer at the bookstore. Each customer needs to make an account in order to track all transactions. Each customer has an ID, full name, and email.
Recommended List:	A list containing recommended books for each customer. Books are added to the list by an employee or bot for customers. Recommended lists have an ID.
Trans:	Any transaction. Each transaction has a unique ID, and they have a customer's ID and total amount.
Wishlist:	A list containing a customer's desired or notable books. A customer adds books to his or her wishlist. Wish lists have a unique ID, and they have a name.

b) The Relational Schema was created from the ER Diagram. The Relational Schema diagram is displayed below, and listed under are each of the tables' functional dependencies.



Functional Dependencies:
Book: {ISBN} -> {Category, Quantity, Title, Year, Price, QuantitySold, Publisher, FirstName, MiddleName, LastName}
Displayed_In: {ISBN, Wishlist} -> {}
Contained_In: {ISBN, TransID} -> {}
Customer: {AccountID} -> {FirstName, LastName, Email}
Trans: {TransID} -> {Date, Downloadable, Shipping, Paymethod, Amount, AccountID}
Held_In: {ISBN, RecommendID} -> {}
Recommended_List: {RecomID} -> {AccountID}
Cart: {BookTitle, AccountID} -> {}
Wishlist: {WishlistID} -> {WishlistName, AccountID}
Contained_In: {ISBN, TransID} -> {}

c) The following list denotes which tables are in Boyce-Codd Normal Form and are not in Boyce-Codd Normal Form.

Book: BCNF
Customer: BCNF
RecommendedList: BCNF
Trans: BCNF
WishList: BCNF
Cart: No Functional Dependencies
Contained_In: No Functional Dependencies
Displayed_In: No Functional Dependencies
Held_In: No Functional Dependencies

The "Cart," "Contained_In," and "Displayed_In" tables do not have any functional dependencies, since they are a product of an N-M relationship. Consequently, they do not have any non-prime attributes. Additionally, each entry can only be uniquely identified by all the prime attributes in each table.
"Book," "Customer," RecommendList," "Trans," and "Wishlist" are all tables in BCNF because

d) Provided here are the different views that were created using our database. For each view, a description is given on what the view represents, and then the respective SQL statements and Relational Algebra statements are also given. A sample output for the views is also found below.

```
CREATE VIEW WishBookAmount(ListID, ListName, AccountID, BookAmount) AS
SELECT W.WishlistID, W.WishlistName, W.AccountID, COUNT(B.ISBN)
FROM Displayed_In AS D, Wishlist AS W, BOOK AS B
WHERE D.WishlistID = W.WishlistID AND B.ISBN = D.ISBN
GROUP BY W.WishlistID, W.WishlistName, W.AccountID;
```

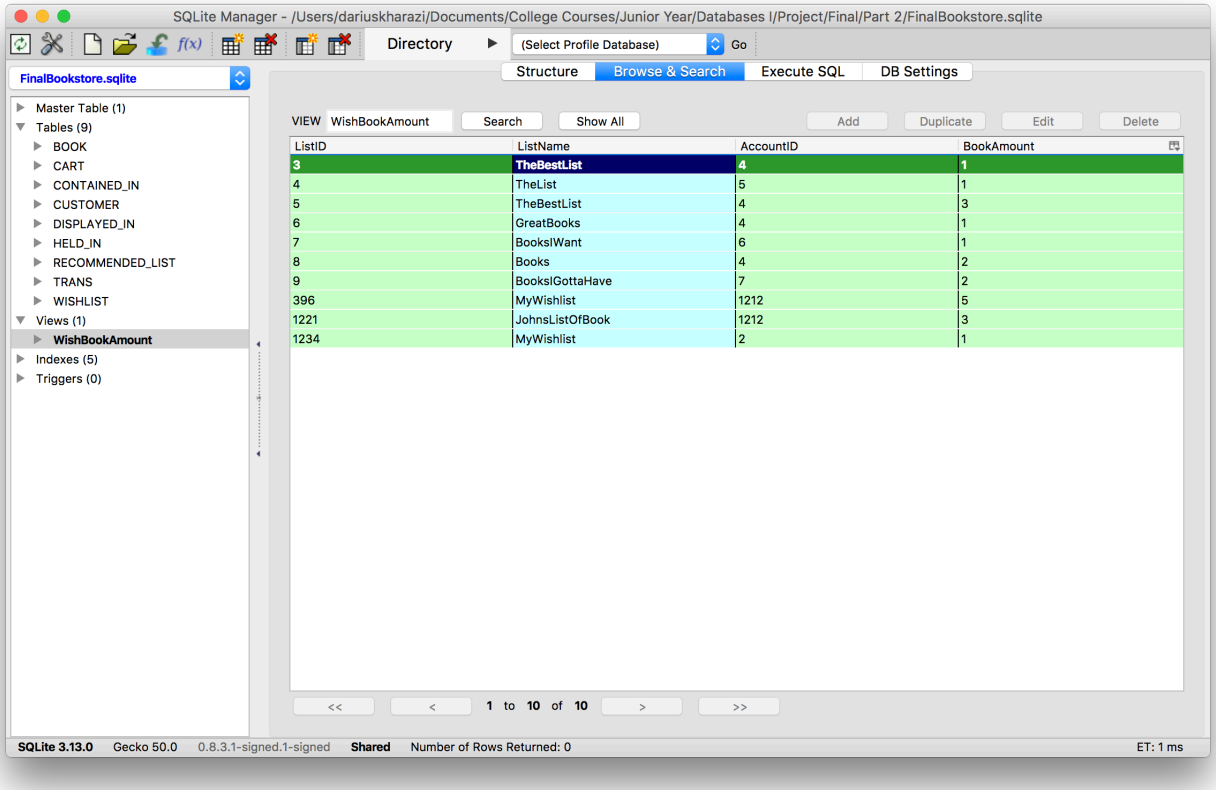
Description: This view would display the amount of books in each wishlist, other useful information about the wishlist, and the ID of the customer who created the wishlist.

Purpose: This view would be useful for the book store's employees. Knowing how many books a user has in his or her wishlist would be helpful for employees who are wanting to send book recommendations or promotions to their customers.

$$\text{TEMP1} \leftarrow \rho_{D}(\text{DISPLAYED\_IN}) \bowtie_{D.ISBN = B.ISBN} \rho_{B}(\text{BOOK})$$

$$\text{TEMP2} \leftarrow \rho_{W}(\text{WISHLIST}) \bowtie_{W.WishlistID = D.WishlistID} \text{Temp1}$$

$$\text{WishBookAmount} \leftarrow {}_{W.WishlistID,\ W.WishlistName,\ W.AccountID}\mathcal{F}_{\text{COUNT B.ISBN}}(\text{TEMP2})$$



```
CREATE VIEW CustPurchases(AccountID, FirstName, LastName, Email, TransTotal, Purchases) AS
SELECT Cu.AccountID, Cu.FirstName, Cu.LastName, Cu.Email, SUM(T.Amount), COUNT(T.TransID)
FROM Trans AS T, Customer AS Cu, Contained_In AS Co
WHERE Cu.AccountID = T.AccountID AND T.TransID = Co.TransID
GROUP BY Cu.AccountID, Cu.FirstName, Cu.LastName, Cu.Email;
```

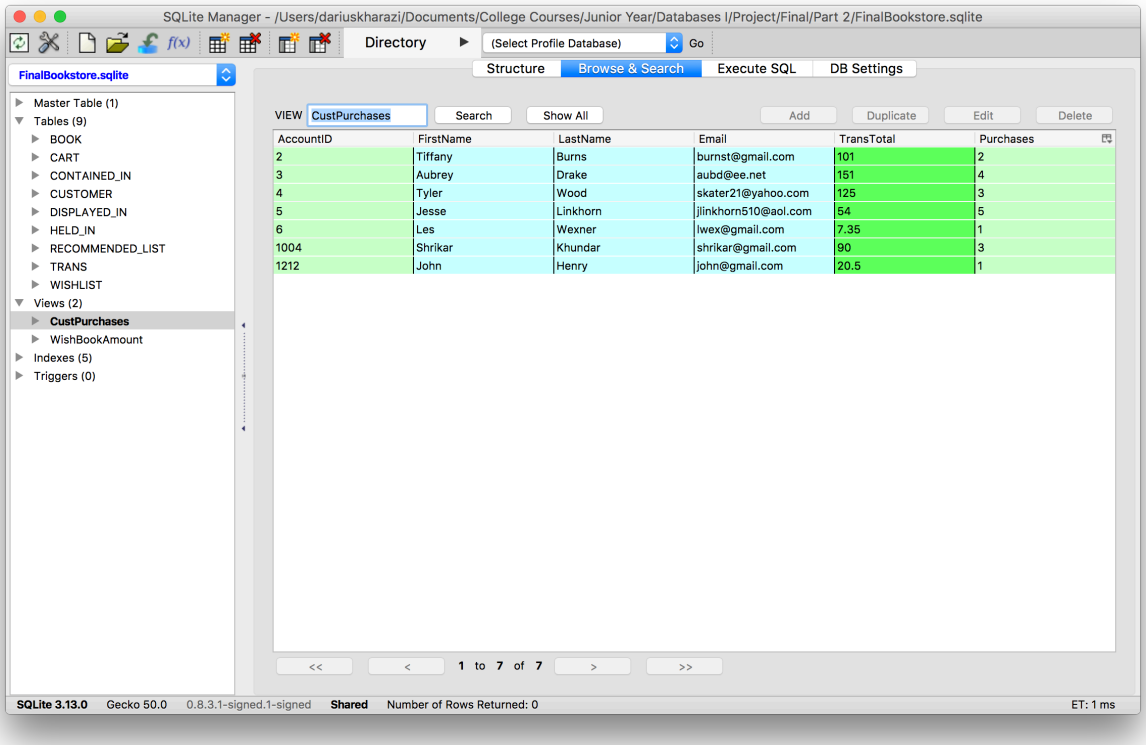
Description: This view would display a complete list of customers, how many books they have purchased, the amount they have spent on purchases, and additional customer information.

Purpose: This view would be extremely useful for employees for marketing purposes. Employees from the book store would be able to monitor each customer's purchases and notice any growth or decline in book purchases from certain customers. If there is any noticeable growth or decline in book purchases, employees could contact the customer by email and send them information about sales, recommended books, etc. Additionally, employees could contact customers about promotions or sales if they see certain customers spending a lot on purchases.

$$\text{TEMP1} \leftarrow \rho_{Cu}(\text{CUSTOMER}) \bowtie_{Cu.AccountID = T.AccountID} \rho_{T}(\text{TRANS})$$

$$\text{TEMP2} \leftarrow \rho_{Co}(\text{CONTAINED\_IN}) \bowtie_{Co.TransID = T.TransID} \text{Temp1}$$

$$\text{CustPurchases} \leftarrow {}_{Cu.AccountID,Cu.FirstName,Cu.LastName,Cu.Email}\mathcal{F}_{\text{COUNT T.TransID,SUM T.AMOUNT}}(\text{TEMP2})$$



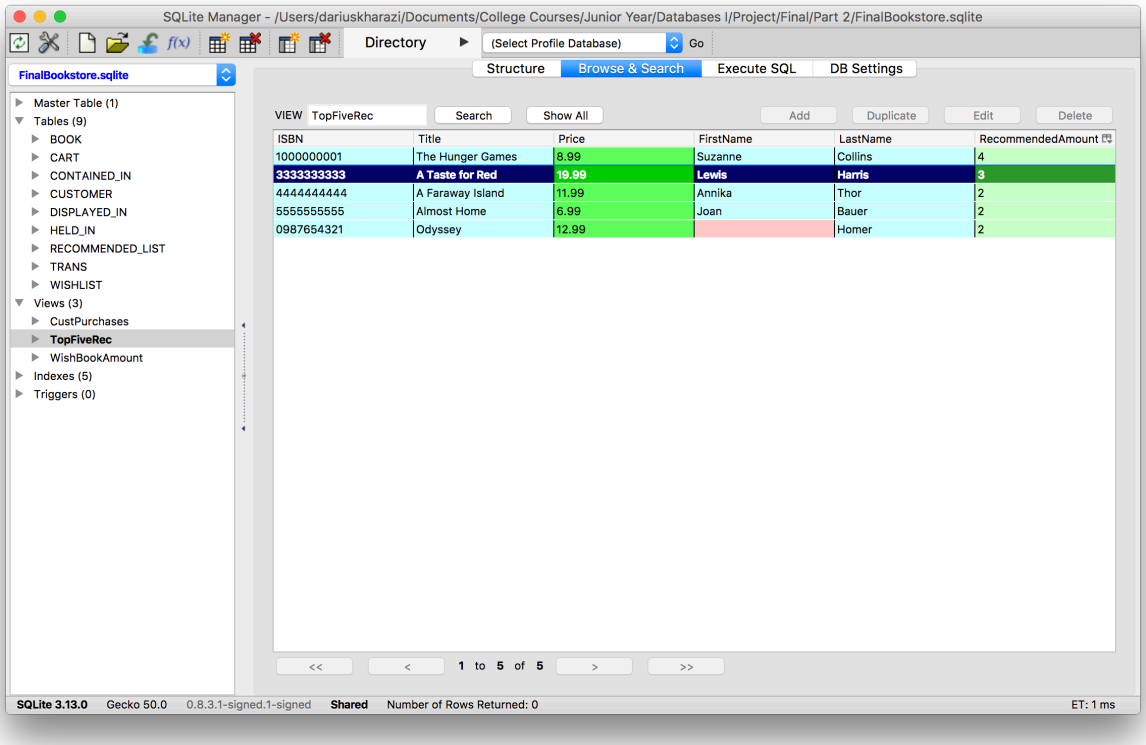
```
CREATE VIEW TopFiveRec(ISBN, Title, Price, FirstName, LastName, RecommendedAmount) AS
SELECT B.ISBN, B.Title, B.Price, B.FirstName, B.LastName, COUNT(B.ISBN)
FROM Book AS B, Held_In AS H
WHERE B.ISBN = H.ISBN
GROUP BY B.ISBN, B.Title, B.Price, B.FirstName, B.LastName
ORDER BY COUNT(B.ISBN) DESC, B.Title ASC
LIMIT 5;
```

Description: This view would display the five most frequently recommended books in the bookstore.

Purpose: This view would be extremely useful for employees from the bookstore, since they could monitor the five most highly recommended books by the bookstore and, perhaps, increase prices accordingly. At times, an author with multiple books might be reoccurring in the top five, which could possibly indicate authors who are trending. This could provide some great insight to the bookstore when promoting sales or other offers.

TEMP1 <- $\rho_{B}(BOOK) \bowtie_{B.ISBN = H.ISBN} \rho_{H} HELD\_IN$

CustPurchases <- $_{B.ISBN,B.Title,\ B.Price,\ B.FirstName,\ B.LastName}\mathcal{F}_{COUNT\ B.ISBN}(TEMP1)$



e) Here are listed two sample Indexes created in our database. A description is given as to what the Index does, as well as a rationale on why we deemed that this was a beneficial Index to create.

Creating tree-based index:

```
CREATE Book_Quantity_Index
ON BOOK(Quantity);
```

Query of Interest:

```
SELECT *
FROM BOOK
WHERE Quantity < 100
```

Description:
The query shown above would enhance the overall performance by creating a tree-based index based on the quantity of each book.

Creating tree-based index:

```
CREATE Book_Year_Index
ON BOOK(Year);
```

Query of Interest:

```
SELECT *
FROM BOOK
WHERE Year < 1995
```

Description:
The query shown below would enhance the overall performance by creating a tree-based index based on the year each book is written.

f) Three sample transactions for our database are given below, with both the SQL code and an accompanying description provided for each:

```
BEGIN;
DELETE FROM CART
WHERE BookTitle in (
        SELECT Title
        FROM Book
        WHERE Price > 20);
COMMIT;
```

This is a transaction that is letting users delete any book from their carts that has a price greater than $20. This is a sequence of read and write operations, as the values are being read from the database to determine which record to delete, and then the corresponding book is deleted from the database.

```
BEGIN;
UPDATE BOOK
SET QuantitySold = QuantitySold +1
WHERE Title = "The Great Gatsby";
COMMIT;
```

This is a transaction that allows the bookstore to increment the quantity sold of a specified book each time there is a new purchase on that book. This is a sequence of read and write operations, since the transaction must first read the value of QuantitySold from the database and then make changes to the value.

```
BEGIN;
UPDATE CUSTOMER
SET Email = "abc123@gmail.com"
WHERE AccountID = 1212;
COMMIT;
```

This is a transaction that allows users to update their email addresses. The sample query enables the user with the AccountID of 1212 to update their email address to abc123@gmail.com. This is also a sequence of read and write operations, because the data of the user must first be read from the database, and then changes to the email are applied.