**Login Page**

"SELECT \* " +

"FROM user " +

"WHERE User\_Login = '" + login + "' " +

"AND User\_Password = '" + password + "';";

If empty, message "Invalid Login or Password. Please try again."

If not, save all 4 outputs, then run following:

"SELECT b.\* " +

"FROM book b, adds\_to\_wishlist w " +

"WHERE w.User\_ID = " + **this**.userID + " " +

"AND w.Book\_ID = b.Book\_ID;";

Save output. Then, for each board run the following:

"SELECT a.\* " +

"FROM author a, is\_written\_by w " +

"WHERE w.Book\_ID = " + **this**.bookID + " " +

"AND w.Author\_ID = a.Author\_ID;";

"SELECT s.\* " +

"FROM series s, is\_part\_of p " +

"WHERE p.Book\_ID = " + **this**.bookID + " " +

"AND p.Series\_ID = s.Series\_ID;";

"SELECT g.Genre\_Name, g.Is\_Tagged\_As\_Number\_Of\_Times " +

"FROM is\_tagged\_as g " +

"WHERE g.Book\_ID = " + **this**.bookID + " " +

"ORDER BY g.Is\_Tagged\_As\_Number\_Of\_Times DESC;"

Save all data. Output of first 2 will by multiple rows. Create string of names/titles (2nd column) for each. I’m not sure what you need to generate pie graph. Ideally, convert the genre information to that format, but at least save the Genre\_Name of the first row, since this is the most common genre. Then proceed to Dashboard.

**Create Account Page**

"SELECT \* " +

"FROM user " +

"WHERE User\_Login = '" + login + "' " +

"AND User\_Password = '" + password + "';";

If not empty, message "Login already exists. Try new login."

If empty, run following:

"INSERT INTO user (User\_Login, User\_Password, User\_Name) " +

"VALUES ('" + login + "', '" + password + "', '" + name + "');";

then

"SELECT User\_ID " +

"FROM user " +

"WHERE User\_Login = '" + login + "';";

Save the 3 inputs and the 1 output, then proceed to Dashboard.

**Dashboard**

If coming from Create Account Page, list should be empty. Else, use information from above to populate table with each book in wishlist (row should have Book Title, string of Author(s), string of Series, string of most popular genre.

If line in wishlist is clicked, go to Book Page for that book. Generate fields from already stored information.

If something is typed in the search bar, and the search bar is pressed, replace all apostrophes in the input with two apostrophes, then:

If dropdown is “Title”:

"SELECT b.\* " +

"FROM book b " +

"WHERE b.Book\_Title LIKE '%" + bookField + "%' " +

"GROUP BY b.Work\_ID " +

"ORDER BY b.Book\_Average\_Rating DESC;";

If dropdown is “Author”:

"SELECT a.\* " +

"FROM author a " +

"WHERE a.Author\_Name LIKE '%" + authorField + "%' " +

"ORDER BY a.Author\_Average\_Rating DESC;";

If dropdown is “Series”

"SELECT s.\* " +

"FROM series s " +

"WHERE s.Series\_Name LIKE '%" + seriesField + "%' " +

"GROUP BY s.Series\_Name;";

Save all information and progress to advanced search/results page for the Book, Author, or Series accordingly.

**Advanced Search – Book**

Clean up the input (get rid of apostrophes from title, author, series, format fields and non-numeric characters from pages and date fields). If min pages is blank, set to 0. If max pages is blank, set to 10000. If min date is blank, set to -3000. If max date is blank, set to 2030. If genre is null,

"SELECT b.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, work w " +

"WHERE b.Book\_Title LIKE '%" + title + "%' " +

"AND a.Author\_Name LIKE '%" + author + "%' " +

"AND s.Series\_Name LIKE '%" + series + "%' " +

"AND b.Book\_Format LIKE '%" + format + "%' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY b.Work\_ID " +

"ORDER BY b.Book\_Average\_Rating DESC;";

else

"SELECT b.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, is\_tagged\_as g, work w " +

"WHERE b.Book\_Title LIKE '%" + title + "%' " +

"AND a.Author\_Name LIKE '%" + author + "%' " +

"AND s.Series\_Name LIKE '%" + series + "%' " +

"AND g.Genre\_Name = '" + genre + "' " +

"AND b.Book\_Format LIKE '%" + format + "%' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Book\_ID = g.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY b.Work\_ID " +

"ORDER BY b.Book\_Average\_Rating DESC;";

Once you have that

SELECT a.Author\_Name

FROM author a, is\_written\_by w

WHERE w.Book\_ID = <Book\_ID>

AND w.Author\_ID = a.Author\_ID

will get you the author(s), to put in the second column

SELECT s.Series\_Name

FROM series s, is\_part\_of p

WHERE p.Book\_ID = <Book\_ID>

AND p.Series\_ID = s.Series\_ID

will get you the series, to put in the third column and

SELECT g.Genre\_Name

FROM is\_tagged\_as g

WHERE g.Book\_ID = <Book\_ID>

ORDER BY g. Is\_Tagged\_As\_Number\_Of\_Times DESC

LIMIT 1

will get you the most common genre to put in the fourth column.

**Advanced Search – Author**

Clean up the input (get rid of apostrophes from name and series fields and non-numeric characters from pages and date fields). If min pages is blank, set to 0. If max pages is blank, set to 10000. If min date is blank, set to -3000. If max date is blank, set to 2030. If genre is null,

"SELECT a.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, work w " +

"WHERE a.Author\_Name LIKE '%" + name + "%' " +

"AND s.Series\_Name LIKE '%" + series + "%' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY a.Author\_ID;";

else

"SELECT a.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, is\_tagged\_as g, work w " +

"WHERE a.Author\_Name LIKE '%" + name + "%' " +

"AND s.Series\_Name LIKE '%" + series + "%' " +

"AND g.Genre\_Name = '" + genre + "' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Book\_ID = g.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY a.Author\_ID;";

**Advanced Search – Series**

Clean up the input (get rid of apostrophes from title and author fields and non-numeric characters from pages and date fields). If min pages is blank, set to 0. If max pages is blank, set to 10000. If min date is blank, set to -3000. If max date is blank, set to 2030. If genre is null,

"SELECT s.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, work w " +

"WHERE s.Series\_Name LIKE '%" + title + "%' " +

"AND a.Author\_Name LIKE '%" + author + "%' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY s.Series\_Name;";

else

"SELECT s.\* " +

"FROM book b, is\_written\_by ba, author a, is\_part\_of bs, series s, is\_tagged\_as g, work w " +

"WHERE s.Series\_Name LIKE '%" + title + "%' " +

"AND a.Author\_Name LIKE '%" + author + "%' " +

"AND g.Genre\_Name = '" + genre + "' " +

"AND b.Book\_Number\_Of\_Pages > " + minPages + " " +

"AND b.Book\_Number\_Of\_Pages < " + maxPages + " " +

"AND w.Work\_Original\_Publish\_Date > " + minDate + " " +

"AND w.Work\_Original\_Publish\_Date < " + maxDate + " " +

"AND s.Series\_ID = bs.Series\_ID " +

"AND bs.Book\_ID = b.Book\_ID " +

"AND b.Book\_ID = g.Book\_ID " +

"AND b.Work\_ID = w.Work\_ID " +

"AND b.Book\_ID = ba.Book\_ID " +

"AND ba.Author\_ID = a.Author\_ID " +

"GROUP BY s.Series\_Name;";

**Addition/Deletion from Wishlist**

If you can set up deletion on the dashboard, that would be ideal, but I couldn’t figure it out in JavaFX. Instead, I made the “Add to Wishlist” buttons on the Book Page say “Remove from Wishlist” if they were already on the user’s wishlist.

To add

"INSERT INTO adds\_to\_wishlist (User\_ID, Book\_ID) " +

"VALUES ('" + **this**.userID + "', '" + curBook.getBookID() + "');";

To delete

"DELETE FROM adds\_to\_wishlist " +

"WHERE User\_ID = " + **this**.userID + " " +

"AND Book\_ID = " + curBook.getBookID() + ";";

**Other**

To find the similar books (“Readers Also Enjoyed”) to a given book:

"SELECT b.\* " +

"FROM is\_similar\_to s, book b " +

"WHERE s.Book\_ID\_1 = " + similarBook.getBookID() + " " +

"AND s.Book\_ID\_2 = b.Book\_ID;";

Load book from Book\_ID

"SELECT b.\* " +

"FROM book b " +

"WHERE b.Book\_ID = " + **this**.bookID + ";";

Load authors from Book\_ID

"SELECT a.\* " +

"FROM author a, is\_written\_by w " +

"WHERE w.Book\_ID = " + **this**.bookID + " " +

"AND w.Author\_ID = a.Author\_ID;";

Load series from Book\_ID

"SELECT s.\* " +

"FROM series s, is\_part\_of p " +

"WHERE p.Book\_ID = " + **this**.bookID + " " +

"AND p.Series\_ID = s.Series\_ID;";

Load genres from Book\_ID

"SELECT g.Genre\_Name, g.Is\_Tagged\_As\_Number\_Of\_Times " +

"FROM is\_tagged\_as g " +

"WHERE g.Book\_ID = " + **this**.bookID + " " +

"ORDER BY g.Is\_Tagged\_As\_Number\_Of\_Times DESC;";

Load work from Book\_ID

"SELECT w.\* " +

"FROM work w " +

"WHERE w.Work\_ID = " + **this**.workID + ";";

Load books from Author\_ID

"SELECT \* " +

"FROM " +

"(SELECT b.\* " +

"FROM book b, is\_written\_by w " +

"WHERE w.Author\_ID = " + **this**.authorID + " " +

"AND w.Book\_ID = b.Book\_ID " +

"GROUP BY b.Work\_ID) AS Works " +

"GROUP BY Book\_Title " +

"ORDER BY Book\_Average\_Rating DESC;"

Load series from Author\_ID

"SELECT s.\* " +

"FROM series s, is\_part\_of p, is\_written\_by w " +

"WHERE w.Author\_ID = " + **this**.authorID + " " +

"AND w.Book\_ID = p.Book\_ID " +

"AND p.Series\_ID = s.Series\_ID " +

"GROUP BY s.Series\_Name;";

Load books from Series\_ID

"SELECT b.\* " +

"FROM book b, is\_part\_of p " +

"WHERE p.Series\_ID = " + **this**.seriesID + " " +

"AND p.Book\_ID = b.Book\_ID";

Load authors from Series\_ID

"SELECT a.\* " +

"FROM author a, is\_written\_by w, is\_part\_of p " +

"WHERE p.Series\_ID = " + **this**.seriesID + " " +

"AND w.Author\_ID = a.Author\_ID " +

"AND w.Book\_ID = p.Book\_ID " +

"GROUP BY Author\_ID;";

Get series average rating from Series\_ID

"SELECT SUM(b.Book\_Average\_Rating \* b.Book\_Ratings\_Count) / SUM(b.Book\_Ratings\_Count) " +

"FROM book b, is\_part\_of p " +

"WHERE p.Series\_ID = " + **this**.seriesID + " " +

"AND p.Book\_ID = b.Book\_ID;";

Check whether book is in user’s wishlist already

"SELECT \* " +

"FROM adds\_to\_wishlist w " +

"WHERE w.User\_ID = " + **this**.userID + " " +

"AND w.Book\_ID = " + curBook.getBookID() + ";";

If not empty, it’s already on the wishlist.