

**Department of Electrical and Computer Engineering**

| Course Number | CPS510 |
| --- | --- |
| Course Title | Database Systems I |
| Semester/Year | F2022 |
| Instructor | Dr. S. B. Tajali |
| Lab Section | 7 |

| Assignment No. | 10 |
| --- | --- |
| Assignment Title | Assignment 10: Submission of Final Documentations |

| Submission Date: | November 27, 2022 |
| --- | --- |
| Due Date: | November 28, 2022 |

| Student Name | Ethan Moo-Young | Ernestine Ho | Abel Abraham Thomas |
| --- | --- | --- | --- |
| Student ID | 501016708 | 501022253 | 501029194 |
| Signature |  |  |  |

Table of Contents

**Phase I - Logical Database Design**

Application Description

Entity-Relationship Model

Schema Design

**Phase II - Implementation**

Database Construction

Simple Queries/Designing Views

Unix Shell Implementation

Normalization

Python GUI

Relational Algebra

**Phase III - Documentation**

Concluding Remarks

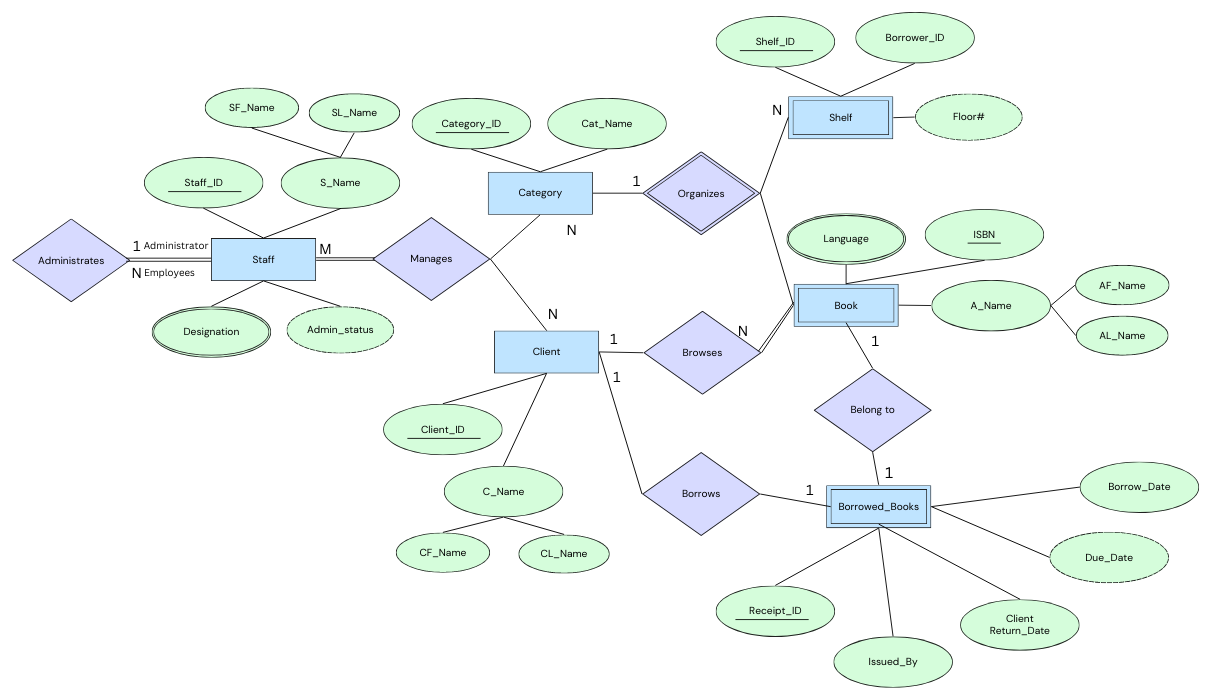
**Phase 1 - Logical Database Design**

## *Application Description:*

Libraries provide a large repository of novels and books for the general public to borrow with a system designed for use by both the librarian and clientele. General users are able to search for various books under their titles, author or category or language and the system is able to return the specific shelf ID and floor location of the specified book. Alternatively, they can search based on category and find the general floor location of the specified genre. A librarian’s main application is to add, remove and alter both client information and books as well as have access to the same functionality as the general users. They are also able to search for outstanding books and the clients they belong to. System Administrators have the added ability to delegate staff roles, identify which staff issues which book and manage all other parts of the system.

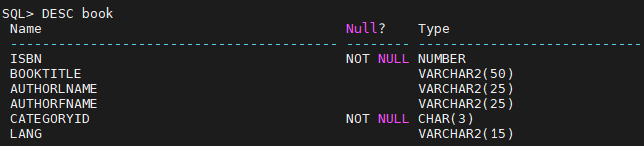
Each BOOK record contains a unique ISBN identifier and corresponds to a specific CLIENT record and CATEGORY record. The CLIENT record is related to a STAFF record. The SHELF record is related to a unique CATEGORY record and gives details on where to find the corresponding book. The BORROWEDBOOKS record keeps track of books that clients have been issued and is related to both the BOOK and CLIENT record Overall, the DBMS we created can support many users and provide many basic functionalities required of a library.

## *Entity Relationship Model*

**

## *Schema Design*

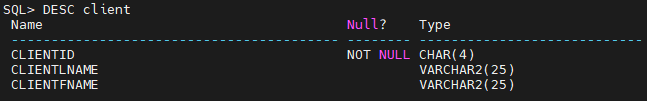
**[Book] Table**



| create table book (  isbn number primary key,  booktitle varchar2(50) default 'Book Title',  authorlname varchar2(25) default 'Last Name',  authorfname varchar2(25) default 'First Name',  categoryid char(3) not null references category(CategoryId),  lang varchar2(15) default 'English'  ); |
| --- |

The [Book] Table entity contains a catalogue of books. Each book includes a unique ISBN identifier as the primary key. The attribute ‘CategoryId’ references other primary keys from other tables and are thus designated as NOT NULL

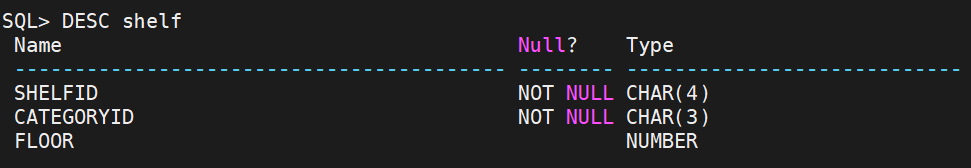
**[Client] Table**



| create table client (  clientid char(4) primary key,  clientlname varchar2(25) default 'Last Name',  clientfname varchar2(25) default 'First Name'  ); |
| --- |

The [Client] Table describes a list of clients associated with the library, its primary key is the ‘ClientId’ attribute.

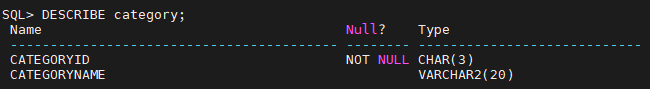
**[Shelf] table**



| create table shelf(  shelfid char(4) primary key,  categoryid char(3) references category(categoryid),  floor number  ); |
| --- |

The [Shelf] Table denotes the specific shelf location and which category it belongs to. The primary key is the ‘ShelfId’ and the ‘CategoryId’ references the [Category] table.

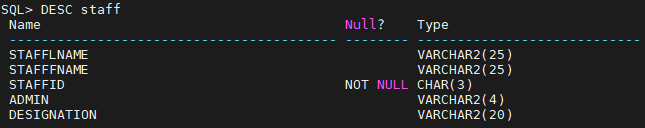
**[Category] Table**



| create table category(  categoryid char(3) primary key,  categoryname varchar2(20)  ); |
| --- |

The [Category] Table describes various categories/genres of book. It is mainly used to organize shelves and books under certain specifications. Its primary key is a specific ‘CategoryId’.

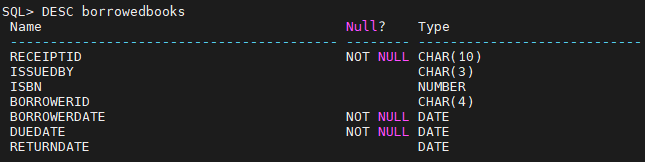
**[Staff] Table**



| create table staff(  stafflname varchar2(25) default 'Last Name',  stafffname varchar2(25) default 'First Name',  staffid char(3) primary key, admin varchar2(4),  designation varchar2(20)  ); |
| --- |

The [Staff] Table describes employees and other staff currently employed by the library. Each Staff has a specific ID as their primary key. Certain staff can also be designated as administrators through the ‘Admin’ attribute as a unique candidate key.

**[BorrowedBooks] Table**



| create table borrowedbooks(  receiptid char(10) primary key,  issuedby char(3) references staff(staffid),  isbn number references book(isbn),  borrowerid char(4) references client(clientid),  borrowerdate date not null,  duedate date not null,  returndate date  ); |
| --- |

The [BorrowedBooks] Table describes books currently borrowed by clients

# **Phase II - Implementation**

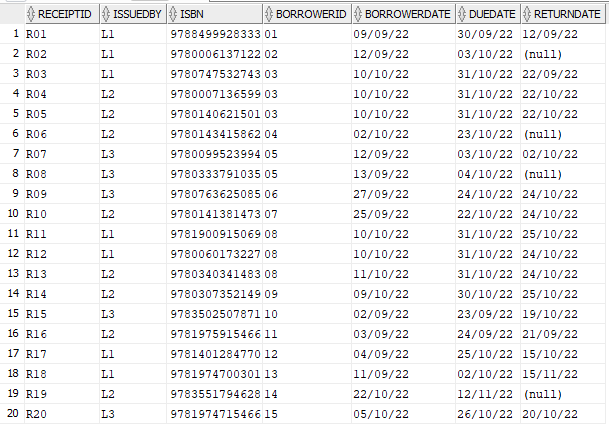
## *Database Contents*

Contents of [Book] Table

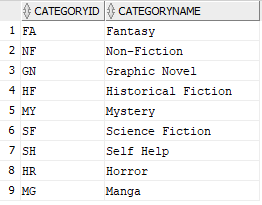




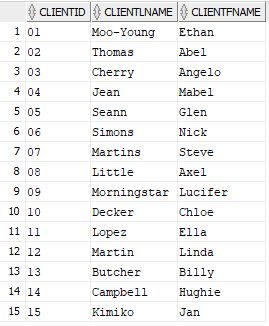
Contents of [BorrowedBooks] Table



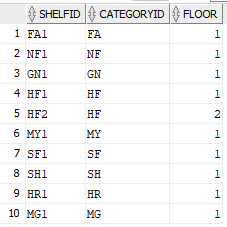
Contents of [Category] Table



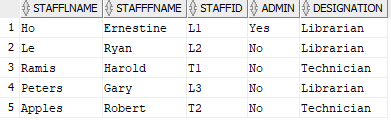
Contents of [Client] Table



Contents of [Shelf] Table



Contents of [Staff] Table



## *Queries/Views*

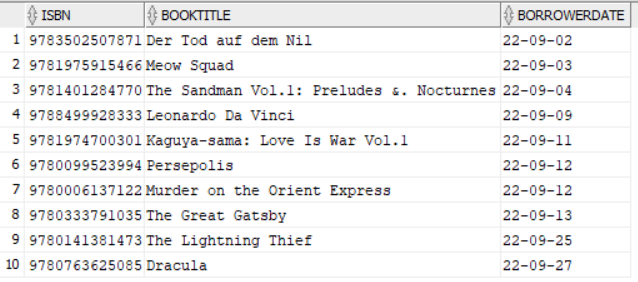
**Queries**

Simple Queries:

Books borrowed by clients in the month of September:

| SELECT b.isbn, b.booktitle, bb.borrowerdate  FROM book b, borrowedbooks bb, client  WHERE bb.borrowerid = client.clientid AND b.isbn = bb.isbn AND EXTRACT(MONTH FROM borrowerdate) = 9 ORDER BY borrowerdate ASC; |
| --- |

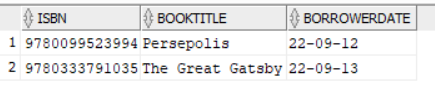
Query Result:



Books borrowed by Glen Seann:

| SELECT b.isbn, b.booktitle, bb.borrowerdate from book b, client c, borrowedbooks bb  WHERE b.isbn = bb.isbn AND bb.borrowerid = '05' AND c.clientid = '05'; |
| --- |

Query Result:



Floor Number of fantasy books:

| SELECT isbn, booktitle, floor FROM book b, shelf sh WHERE b.categoryid = 'FA' AND b.categoryid = sh.categoryid ORDER BY booktitle ASC; |
| --- |

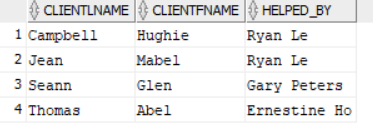
Query Result:



Select Distinct clients with borrowed books and which staff helped the client:

| SELECT DISTINCT clientlname, clientfname, stafffname || ' ' || stafflname AS helped\_by  FROM client c, book b, staff s, borrowedbooks bb  WHERE bb.returndate IS NULL AND bb.borrowerid = c.clientid AND bb.issuedby = s.staffid; |
| --- |

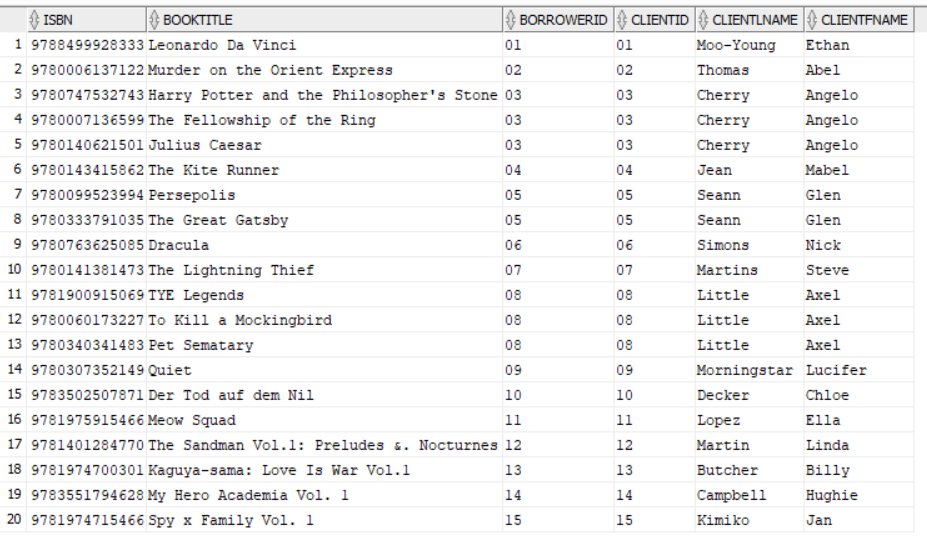
Query Result:



Books that have been borrowed by a client:

| SELECT book.isbn, book.booktitle, borrowedbooks.borrowerid, client.clientid, client.clientlname, client.clientfname from book, borrowedbooks, client WHERE book.isbn = borrowedbooks.isbn AND borrowedbooks.borrowerid = client.clientid ORDER BY clientid; |
| --- |

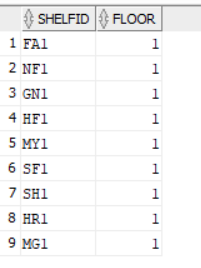
Query Result:



Select all the shelfs where floor is 1:

| SELECT shelfid, floor  FROM shelf  WHERE floor = '1'; |
| --- |

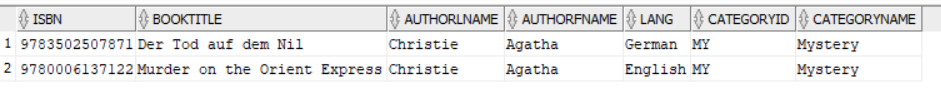
Query Result:



Books by author Agatha Christie:

| SELECT isbn, booktitle, authorlname, authorfname, lang, book.categoryid, categoryname  FROM book, category  WHERE book.categoryid = category.categoryid AND authorlname = 'Christie' AND authorfname = 'Agatha'; |
| --- |

Query Result:



LIbrarian issuing book/books to a client (with the book and borrowing date):

| SELECT receiptid,issuedby, clientid, clientlname, clientfname, booktitle, borrowerdate  FROM borrowedbooks bb, client c, book b  WHERE bb.borrowerid = c.clientid AND b.isbn = bb.isbn ORDER BY receiptid; |
| --- |

Query Result:

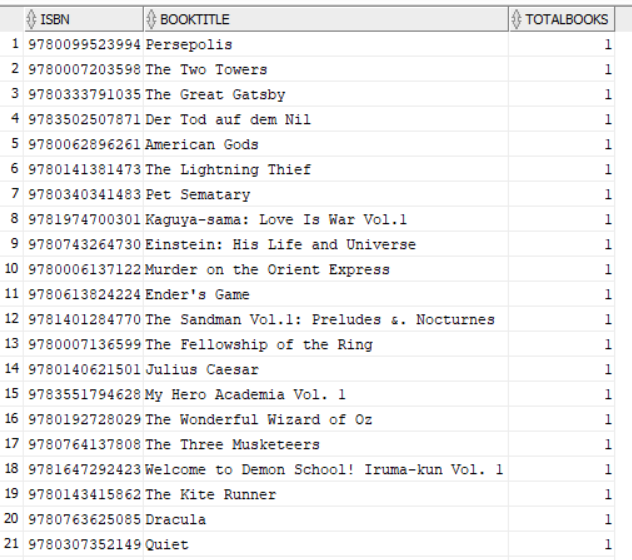


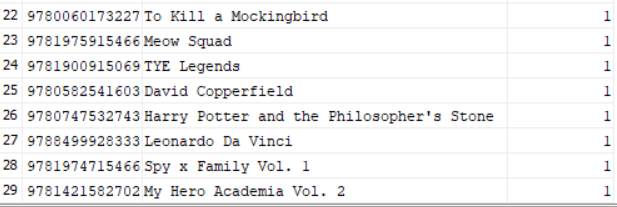
Advanced Queries:

Counting all the books:

| SELECT DISTINCT isbn, booktitle, COUNT (\*) AS totalbooks  FROM book  WHERE isbn IS NOT NULL  GROUP BY isbn, booktitle; |
| --- |

Query Result:

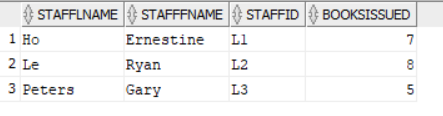




Number of books issued by each librarian:

| SELECT stafflname, stafffname, staffid, COUNT (b.issuedby) AS booksissued FROM client c, staff, borrowedbooks b WHERE c.clientid = b.borrowerid AND b.issuedby = staff.staffid GROUP BY stafflname, stafffname, staffid; |
| --- |

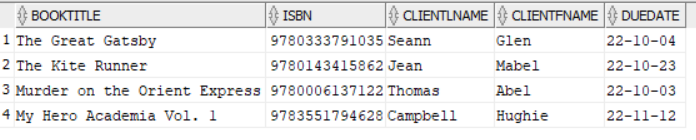
Query Result:



Books borrowed from clients who have not returned their books:

| SELECT b.booktitle, b.isbn, clientlname, clientfname, duedate FROM book b, client c, borrowedbooks bb WHERE returndate IS NULL AND b.isbn = bb.isbn AND c.clientid = bb.borrowerid; |
| --- |

Query Result:



How long it takes for clients to return books:

| SELECT clientid, clientlname, clientfname, AVG(returndate - borrowerdate) FROM client c, borrowedbooks b WHERE c.clientid = b.borrowerid GROUP BY clientid, clientlname, clientfname HAVING AVG(returndate - borrowerdate) > (SELECT AVG(returndate - borrowerdate) FROM client, borrowedbooks ); |
| --- |

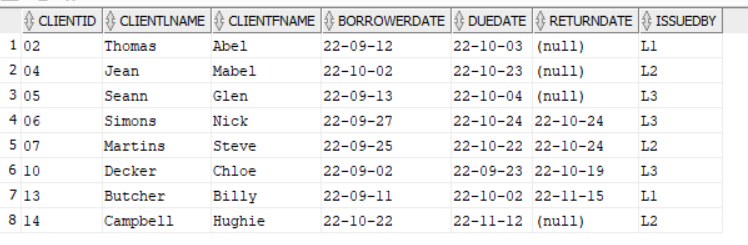
Query Result:



Clients who have not returned their books and returned their books late:

| SELECT clientid, clientlname, clientfname, borrowerdate, duedate, returndate, issuedby  FROM client c, borrowedbooks b  WHERE b.borrowerid = c.clientid AND EXISTS (SELECT clientid, clientlname, clientfname, borrowerdate, duedate, returndate, issuedby FROM client , borrowedbooks WHERE returndate IS NOT NULL) MINUS (SELECT clientid, clientlname, clientfname, borrowerdate, duedate, returndate, issuedby  FROM client, borrowedbooks WHERE returndate < duedate); |
| --- |

Query Result:

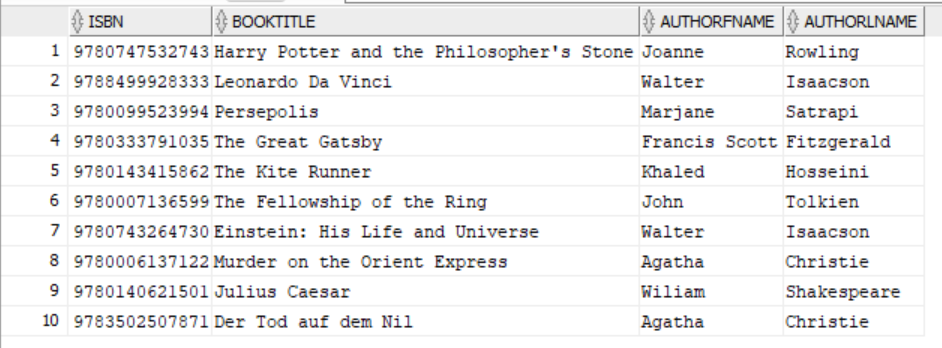


**Views**:

A View table for Client to check what books are available in the library:

| CREATE VIEW available\_books AS  (SELECT isbn, booktitle, authorfname, authorlname  FROM book  WHERE booktitle IS NOT NULL); |
| --- |

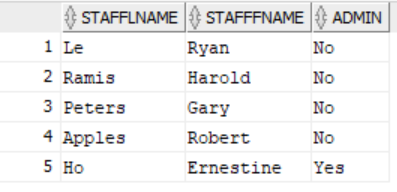
Views table Result:



A View table for owner to see which staff has admin status:

| CREATE VIEW staff\_admin\_status AS  (SELECT stafflname, stafffname, admin  FROM staff  WHERE stafflname IS NOT NULL AND stafffname IS NOT NULL); |
| --- |

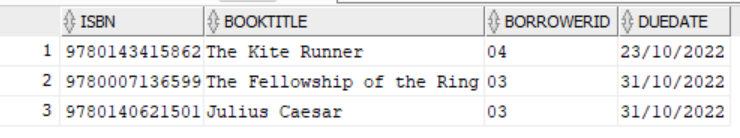
Views table Result:



A View table for staff to see which books are not returned yet:

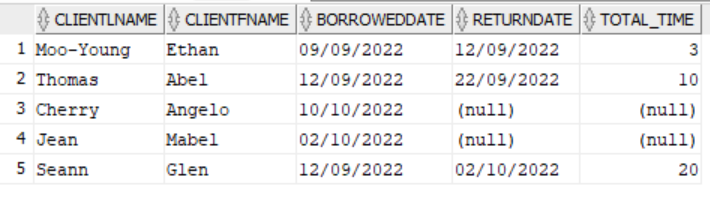
| CREATE VIEW unreturned\_books AS  (SELECT isbn, booktitle, borrowerid, duedate  FROM book, client  WHERE client.returndate IS NULL AND book.borrowerid = client.clientid); |
| --- |

Views table Result:



A View table for staff to see statistics of time clients takes to return book after borrowing:

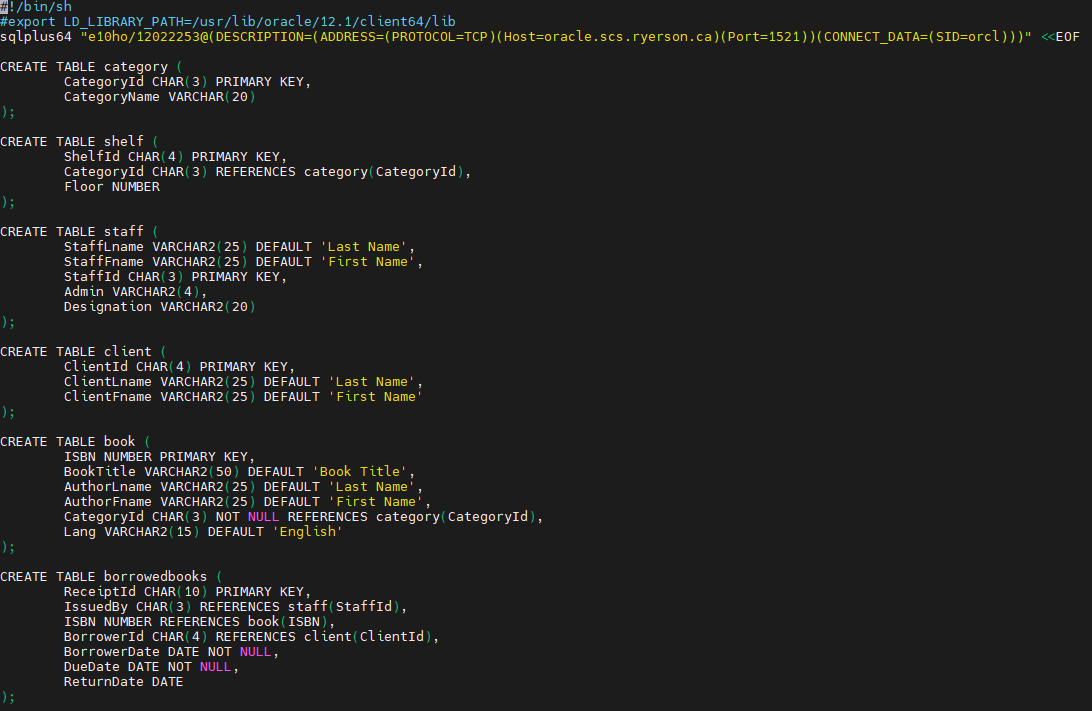
| CREATE VIEW client\_borrowtime AS  (SELECT clientlname, clientfname, borroweddate, returndate, ROUND(returndate - borroweddate) AS total\_time  FROM client  WHERE clientlname IS NOT NULL AND clientfname IS NOT NULL); |
| --- |

Views table Result:  


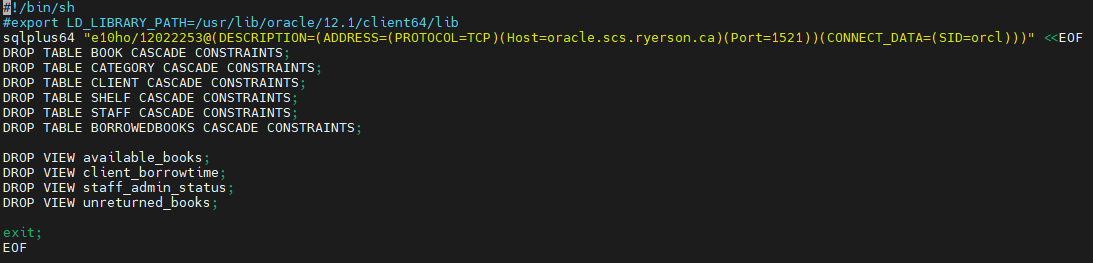
## *Unix Shell Implementation*



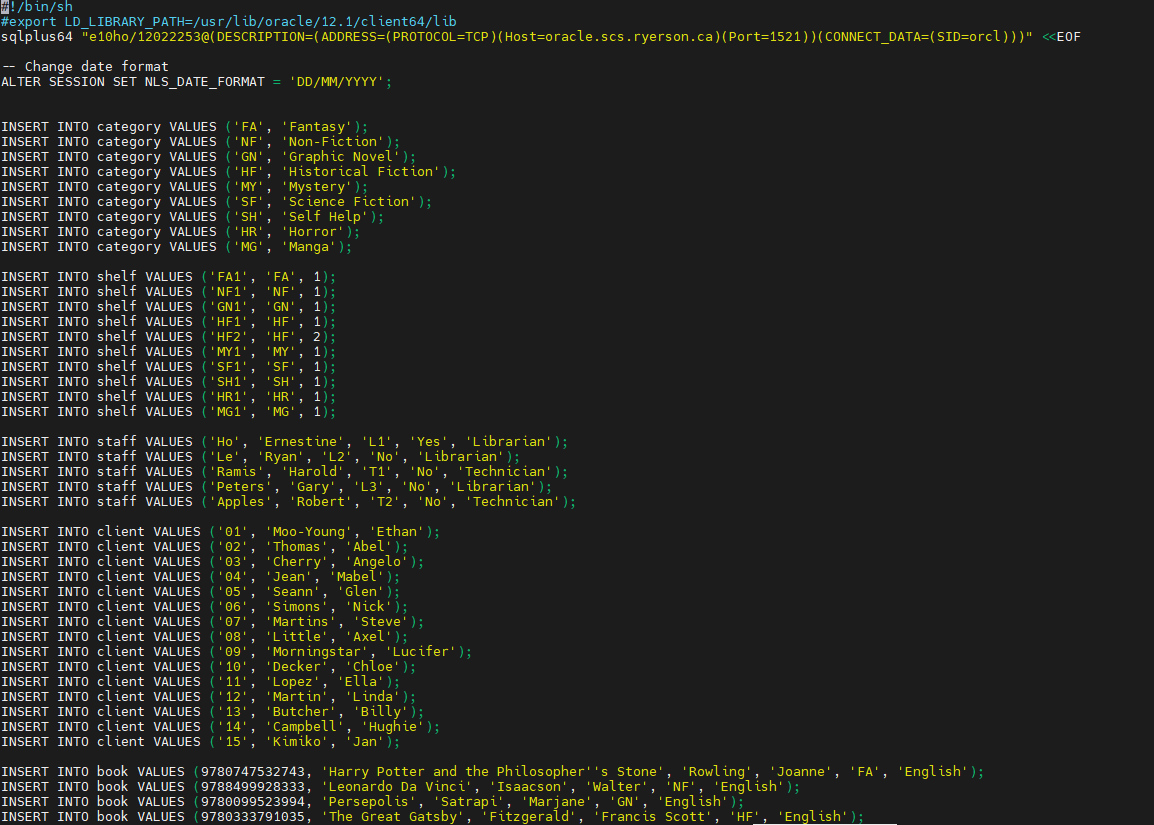
create\_tables.sh

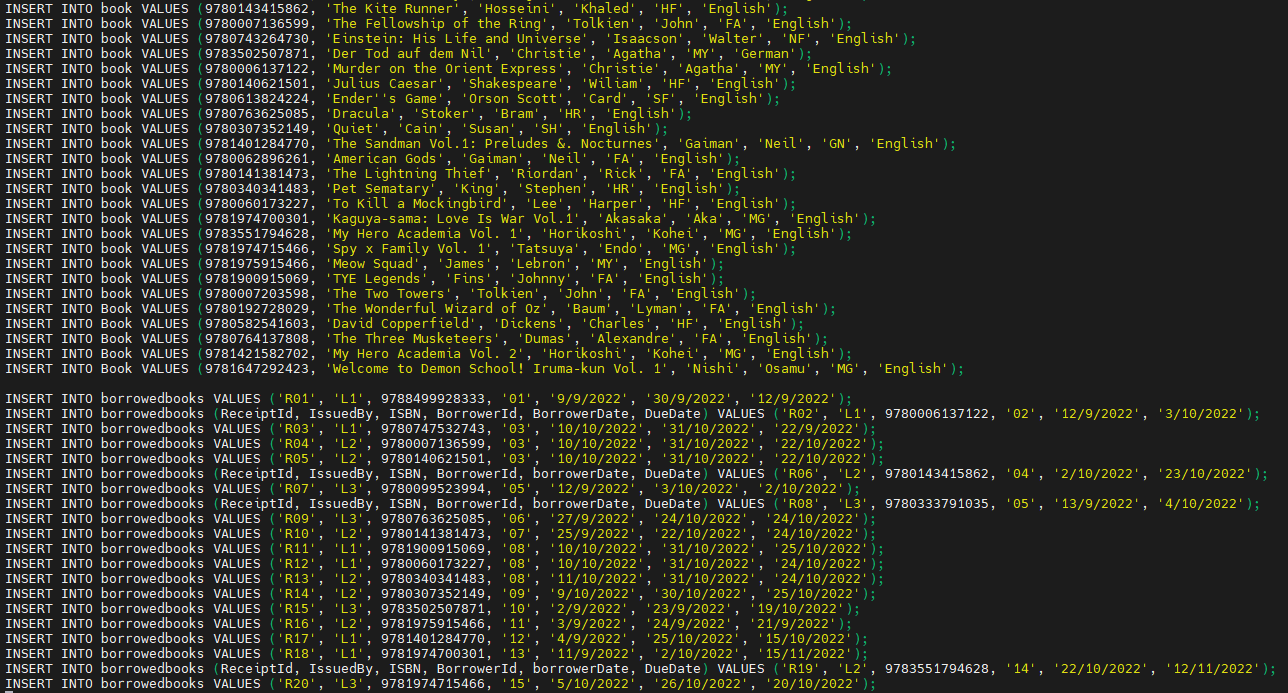


drop\_tables.sh

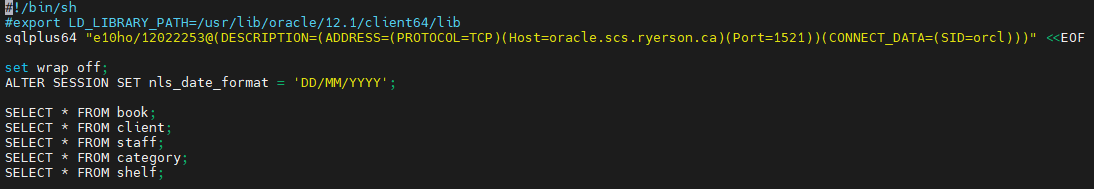


populate\_tables.sh

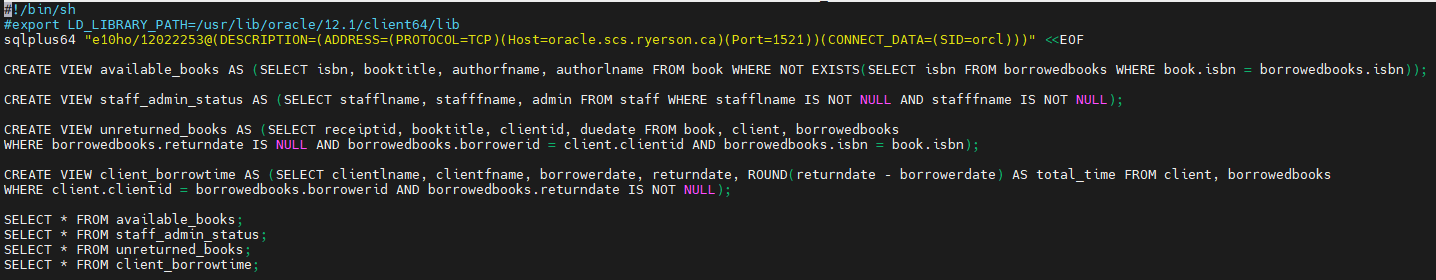




queries.sh



view\_table.sh



## *Normalization*

Relational schema before normalization:

Book(ISBN, BookTItle, AuthorLName, AuthorFName, CategoryId, Lang, ShelfId, BorrowerId)

Category(CategoryId, CategoryName)

Client(ClientId, ClientLName, ClientFName, BorrowedDate, DueDate, ReturnDate, IssuedBy)

Shelf(ShelfId, CategoryId, Floor)

Staff(StaffLName, StaffFName, StaffId, Admin, Designation)

Final relational schema:

Book(ISBN -> BookTitle, AuthorLname, AuthorFname, CategoryId, Lang)

BorrowedBooks(ReceiptID -> IssuedBy, ISBN, BorrowerID, BorrowerDate, DueDate)

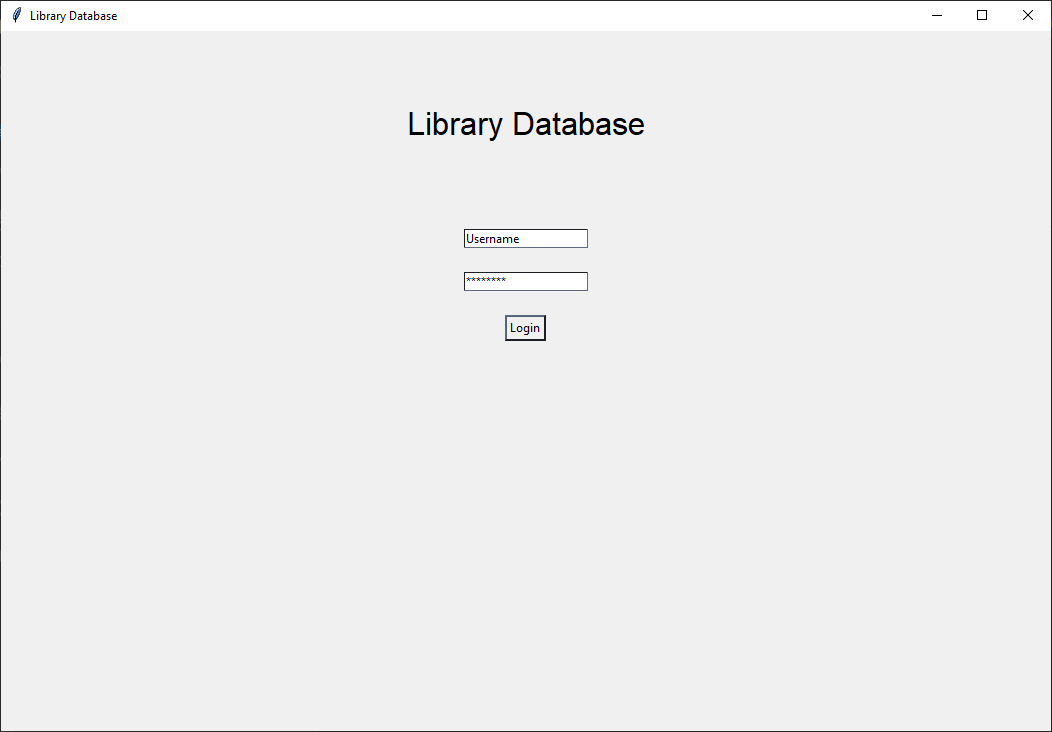
Category(CategoryId -> CategoryName)

Client(ClientId -> ClientLname, ClientFname)

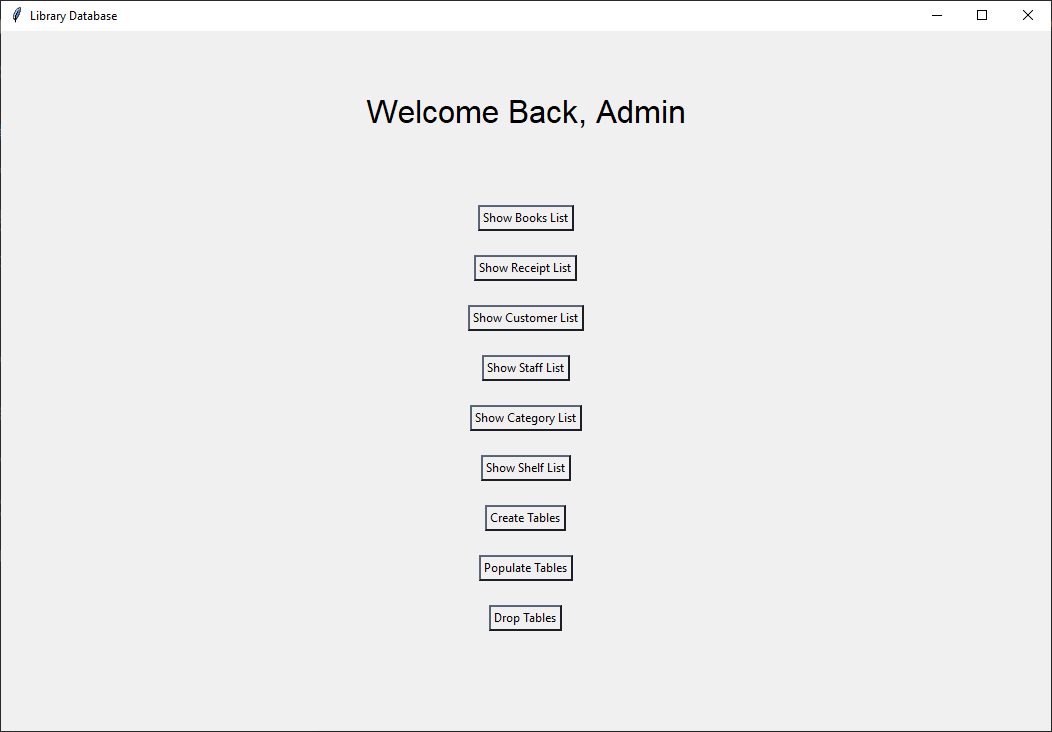
Shelf(ShelfId -> CategoryId, Floor)

Staff(StaffId -> StaffLname, StaffFname, Admin, Designation)

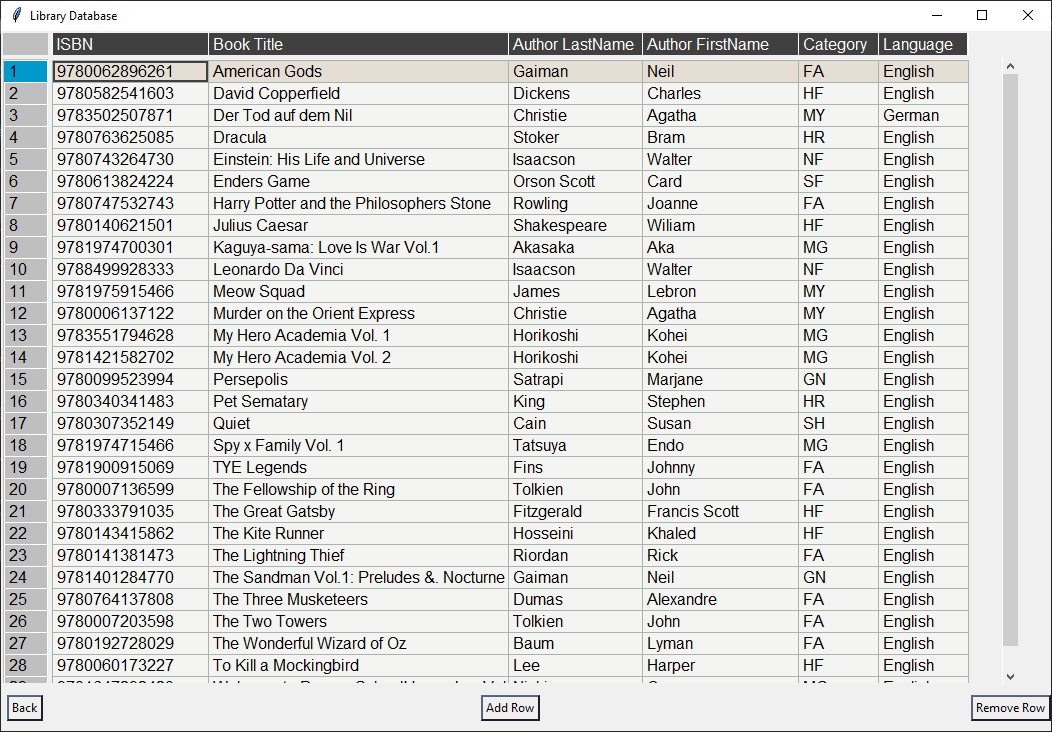
## *Python GUI*



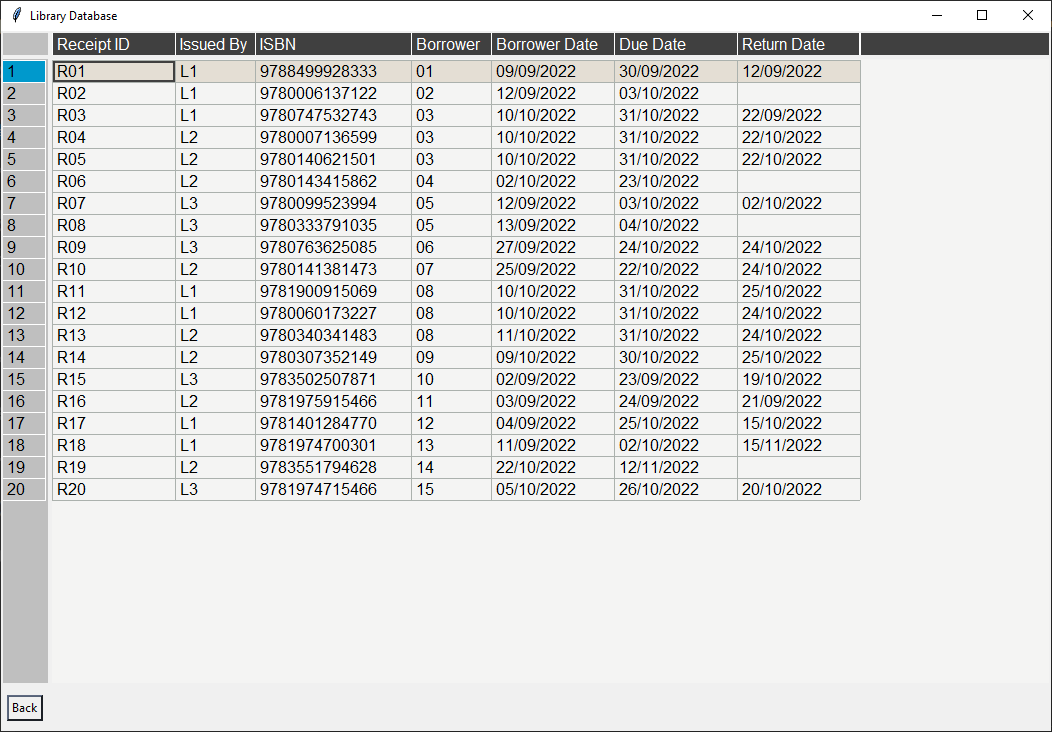
**Login Page**



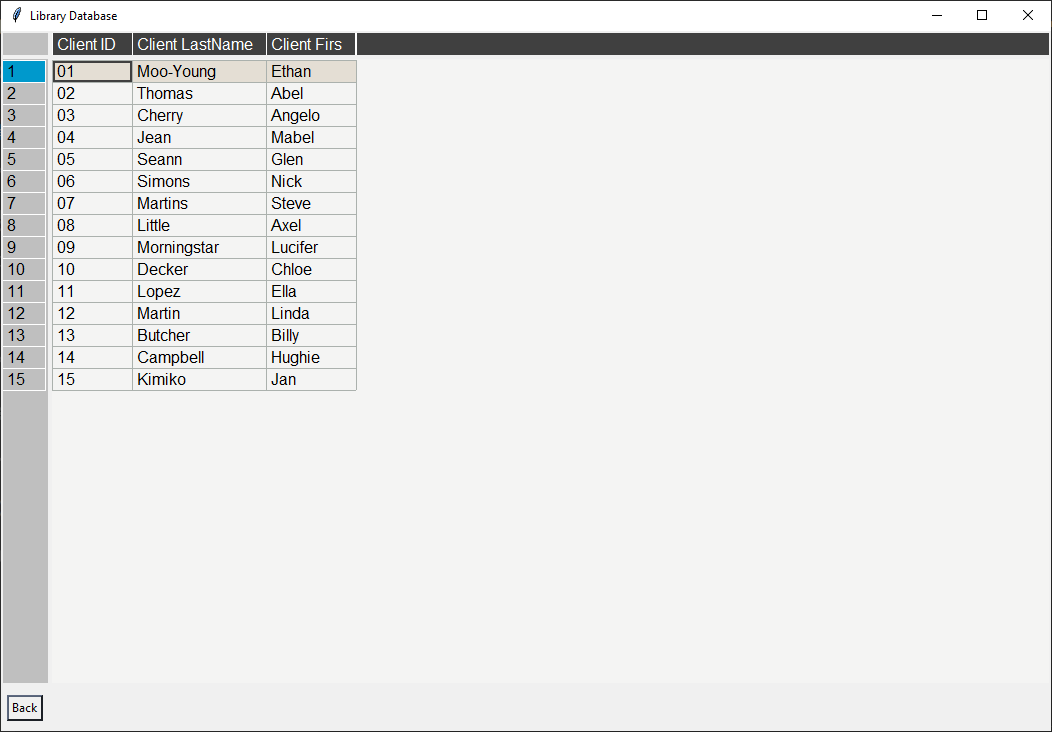
**Welcome Page**



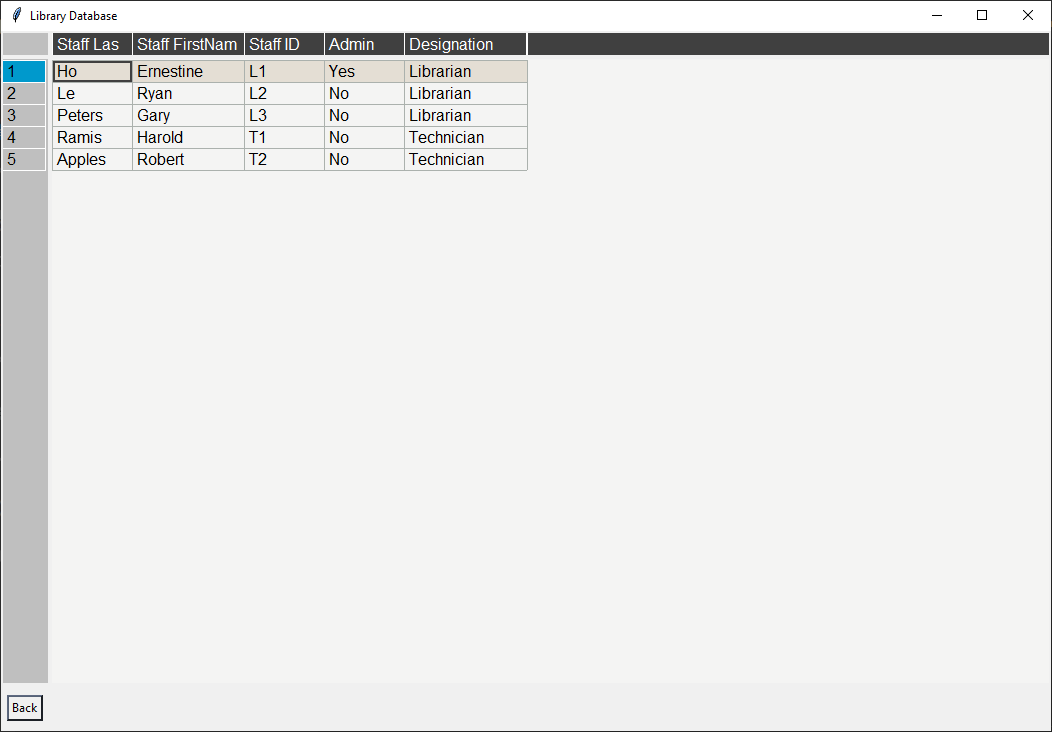
**Book Table**



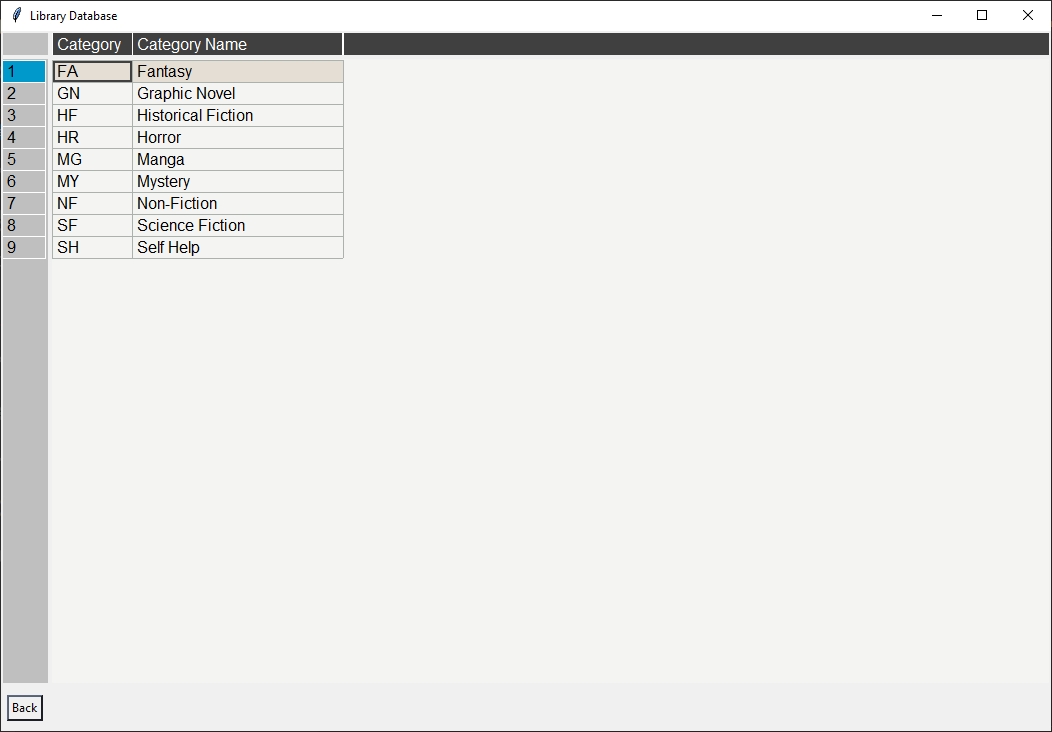
**Receipt/Borrowed Books Table**



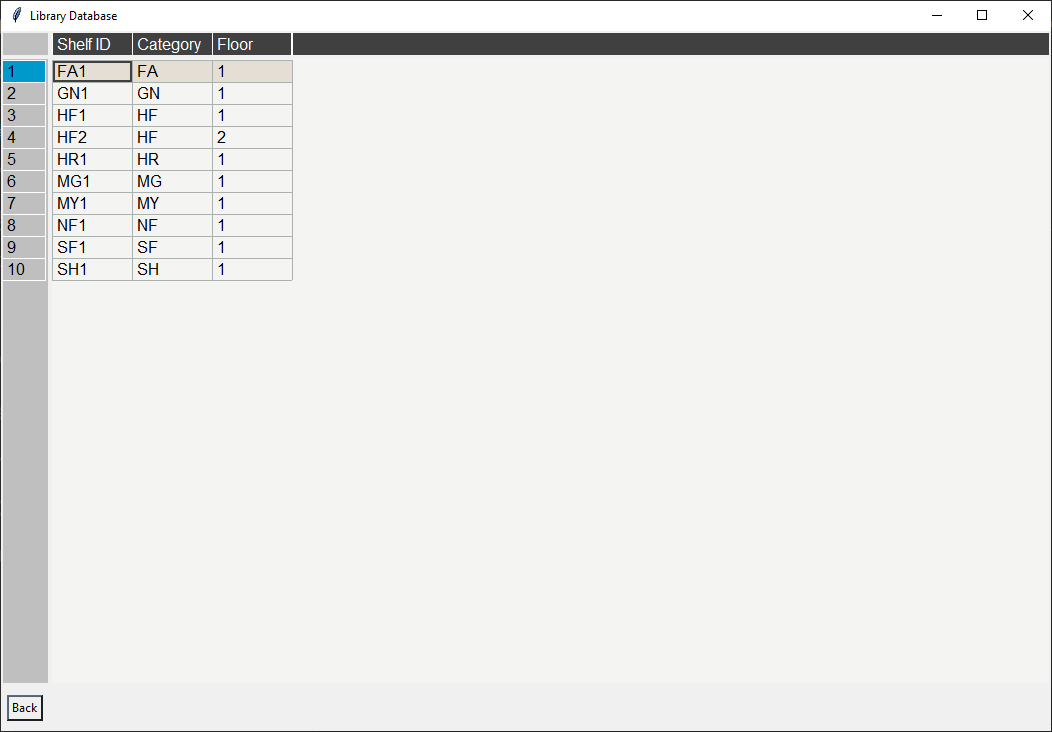
**Client Table**



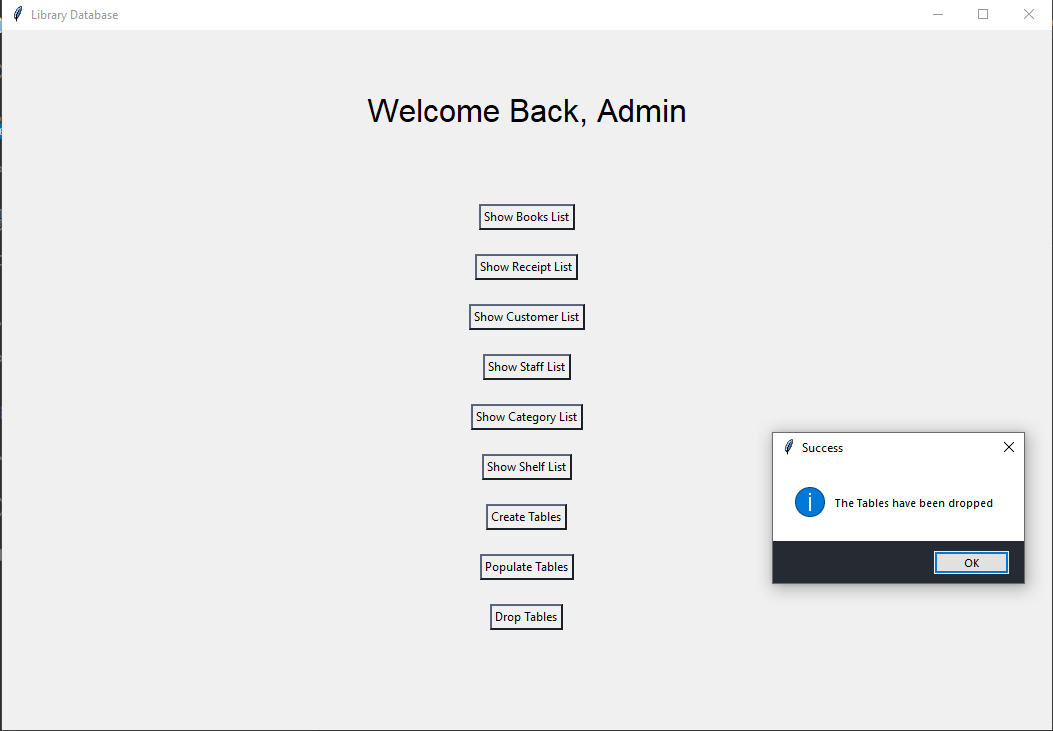
**Staff Table**



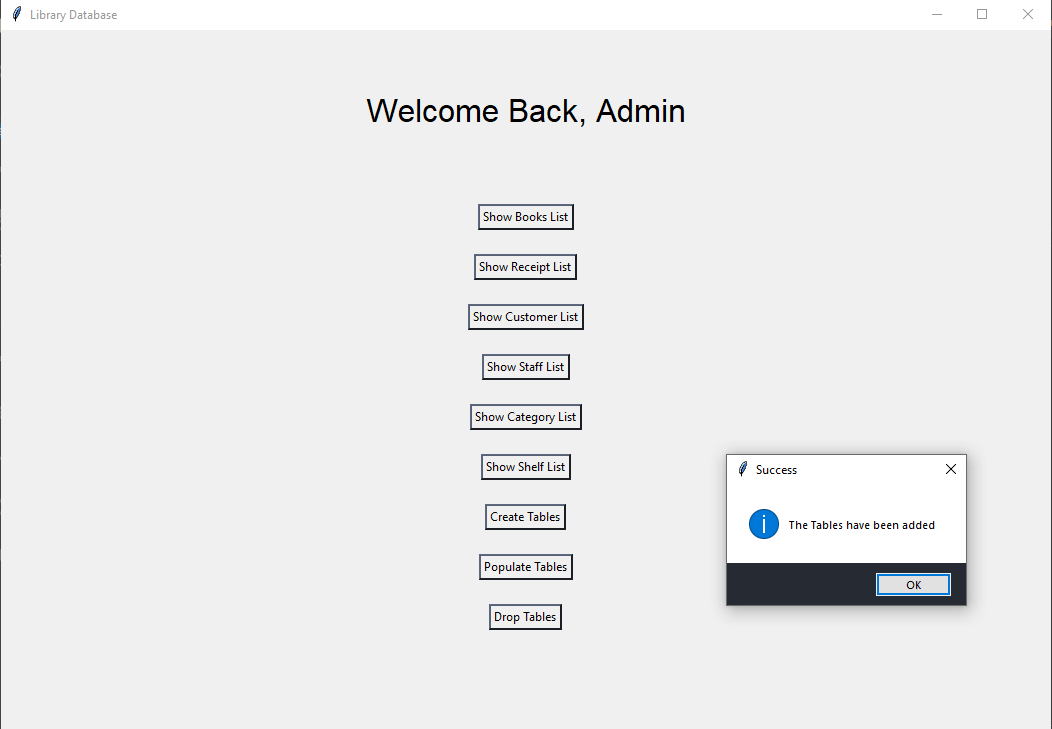
**Category Table**



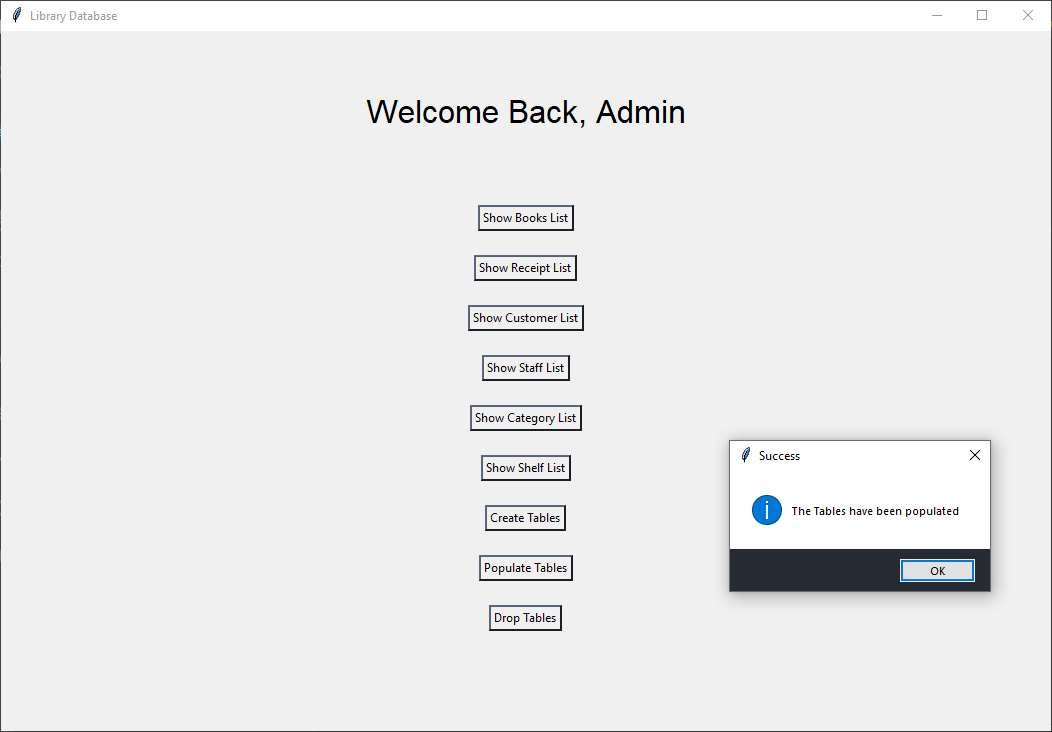
**Shelf Table**



**Dropping Table**



**Add Table**



**Populate Table**

## *Relational Algebra*

1. Books borrowed by clients in the month of September

| (BorrowedBooks) |
| --- |

1. Books borrowed by Glen Seann

| (BorrowedBooks) |
| --- |

1. Floor Number of Fantasy books

| ((Shelf ▷◁ Category)) |
| --- |

1. Distinct clients with borrowed books and which staff helped the client

| ((Client ▷◁ BorrowedBooks)) |
| --- |

1. Books that have been borrowed by a client

| (BorrowedBooks) |
| --- |

1. Counting all the books

| (Books) |
| --- |

1. Number of books issued by each librarian

| (BorrowedBooks ▷◁ Staff) |
| --- |

1. Books borrowed from clients who have not returned their books

| ((Client ▷◁ BorrowedBooks ▷◁ Books)) |
| --- |

1. How long it takes for clients to return books

| (Books) |
| --- |

1. Clients who have not returned their books and returned their books late

| (((Client ▷◁ BorrowedBooks)))∪ (((Client ▷◁ BorrowedBooks))) |
| --- |

# 

# **Phase III - Documentation**

## *Concluding Remarks*

In conclusion, after designing this database I was better able to understand the reason behind a database. The database helped us store a large volume of data tables all in one place. It is basically a computerized record- keeping system just as mentioned in lecture 1. We started off by choosing the library as our database system which we later then created an ER diagram. The ER diagram was crucial as it was the foundation of our database design. In Particular, it showed the big picture of depicting the relationships within the DBMS. The schema part was interesting as it actually put the ER diagram into words which has then been used to implement our database system on Oracle. The queries and views were the most important, challenging and satisfying part of the database. I truly enjoyed this part of the assignment, updating the queries/views after the normalization was an emotional rollercoaster. Despite the challenges it truly focused on understanding and applying the relationships between the tables. The normalization. relational algebra and the GUI implementation was equally challenging but was doable. It was crucial for a well maintained DBMS. Overall, after all the challenges faced throughout the assignments, it was well worth the final goal of the course. It truly met the requirements from the course outline. I learned a lot about DBMS, its properties and applications. Thus, proving a successful DBMS project.