**ASSIGNMENT 2 FRONT SHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification** | **TEC Level 5 HND Diploma in Computing** | | |
| **Unit number and title** | **Unit 04: Database Design & Development** | | |
| **Submission date** | 3/5/2022 | **Date Received 1st submission** |  |
| **Re-submission Date** |  | **Date Received 2nd submission** |  |
| **Student Name** | Nguyen Tran Nam Khanh | **Student ID** | GCH210731 |
| **Class** | GCH1003 | **Assessor name** | Dang Tran Long |
| **Student declaration**  I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice. | | | |
|  |  | **Student’s signature** |  |

**Grading grid**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P5 | M2 | M3 | M4 | M5 | D2 | D3 |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **❒ Summative Feedback: ❒ Resubmission Feedback:** | | |
| **Grade:** | **Assessor Signature:** | **Date:** |
| **Signature & Date:** | | |

Table of Contents

[I. Develop the database system 5](#_Toc102458252)

[1. Queries to create database and tables (and related elements needed) 5](#_Toc102458253)

[A. Creating different tables 5](#_Toc102458254)

[B. Adding constraint (primary key & foreign key) 10](#_Toc102458255)

[C. Final ERD 11](#_Toc102458256)

[D. Database diagram 12](#_Toc102458257)

[E. Mock-up 14](#_Toc102458258)

[II. Produce Queries 23](#_Toc102458259)

[1. Queries to INSERT data with illustrations 23](#_Toc102458260)

[2. Queries to UPDATE data 25](#_Toc102458261)

[3. Queries to DELETE data 28](#_Toc102458262)

[4. Queries to SELECT data 29](#_Toc102458263)

[5. Advanced queries + Evaluation on effectiveness of the systems 34](#_Toc102458264)

[A. Create view and index 34](#_Toc102458265)

[B. Insert trigger 35](#_Toc102458266)

[C. Login procedure 36](#_Toc102458267)

[D. Evaluation on effectiveness of the systems 37](#_Toc102458268)

[III. Test the system 37](#_Toc102458269)

[1. Test cases 37](#_Toc102458270)

[2. Flowcharts of the system 38](#_Toc102458271)

[A. Update (pay fine) 38](#_Toc102458272)

[B. Insert 40](#_Toc102458273)

[C. Delete 41](#_Toc102458274)

[D. Login 42](#_Toc102458275)

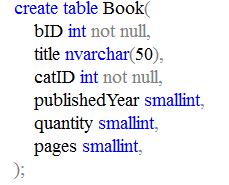
# Develop the database system

## Queries to create database and tables (and related elements needed)

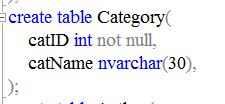
* This assignment will demonstrate different queries which is essential to develop the database system that is based on the scenario that was given. In addition, a user-interface as known as mock up will be presented based on the design of the database.
* The next section will display the process of creating all the tables needed in the database. The constraint will be presented to ensure the unique of the data and the link between tables. ERD of the database and the final mockup will also be given.
* Since this is an improvement of the last assignment, the aspect of creating tables and adding keys had been covered. For that reason, I will take this section directly from my assignment 1.

### Creating different tables

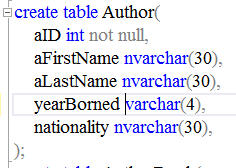
* Since this is an Online Library System which is used to manage a large data of books, the core of the system should be the books. The first thing I create is the book data which contains the IDs, name, etc. The following screenshot is taken from my source code as it demonstrates the creating of the book table as well as the data inside it.



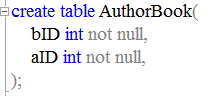
* There are 5 different elements in the table which means every book in the database will contain that 5 different elements. Those are their IDs, names, category ID, published year, the copies that are in the stock and the number of pages in the book.
* In more details, the crucial part of the table is the ID which is set to be unique so it must be not null.
* The title of the book is set to be nvarchar instead of varchar since there will be a lot of non-English books that need their title to have unique characters. I set the limit of this to 50 because there are some books that have very long title.
* Category ID is the same as the book ID and this will be used to connect to the Category table since books can have the same category. As for the published year, quantity and pages, these are positive number and the rarely go too large so smallint should be enough to make the database the most utilized.
* Since we have Category ID, it is suitable to have a category table for this type of data.



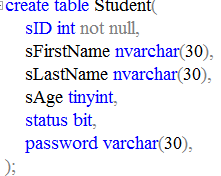
* This table only serve the purpose of bridging the category names to their ids so the structure is not complex. catID for the ID which is the same as the one on the Book table and the catName is for theirs name with the limit of 30 characters.
* The next table that I created is the Author table:



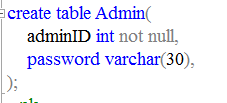
* The authors will have their own ID as well and these cannot be null for crucial reason. As for the names it is fairly reasonable to split them into first name and last name just for more impactful data. The limitation of 30 characters should be enough to cover some person with longer name. In addition, the problem of authors having the same names is likely to happen. However, this is resolved by simply having their own unique IDs. The later information (Year of birth, nationality) is for more details of the person since the viewer sometimes look up for those. The Year of birth data is set to be varchar of 4 which is suitable for someone year of birth as well as the nationality to be set for nvarchar 30.
* The Book and the Author table is two different entities so they need to have a link. This is when AuthorBook is introduced. Since there are not relation and information in the linking, the structure is also simple.



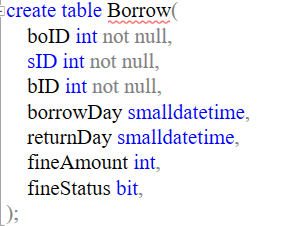
* There are two data here which is BookID and the AuthorID, both are int not null like their twins in the Book and Author tables.
* The next table is the Student or Member of the library table:



* The studentID and their name similar to that of author. Student’s age is set to be tinyint since the age of one are usually not exceed 100. Status of bit is normal for data type like this only have two options. Finally, student will have password so they can be separate from admin.

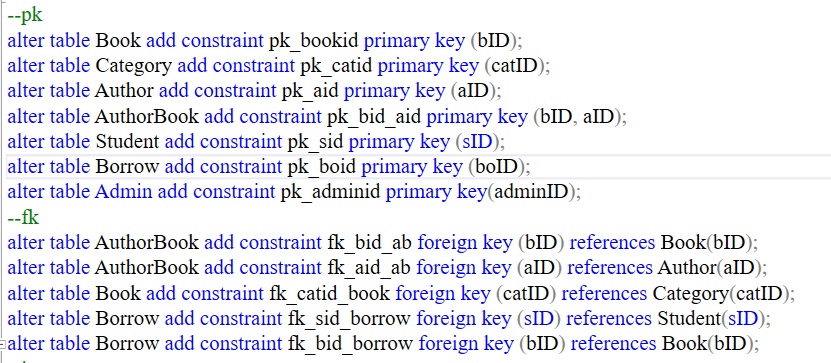


* As I was mentioned, there is an admin for verifying the admin and student.
* The next table is borrow table for borrowing book:



* boID is for borrow ID which is the same function as the other IDs. This table will be connect will Student table and Book table since a loaned book will have a borrower and which is borrowed. There is a date next, which is the day of borrowing. This will be crucial when combining with return day for finding out whether or not student need to pay fine. The fine status is determined by the admin if the student pay the fine or not.

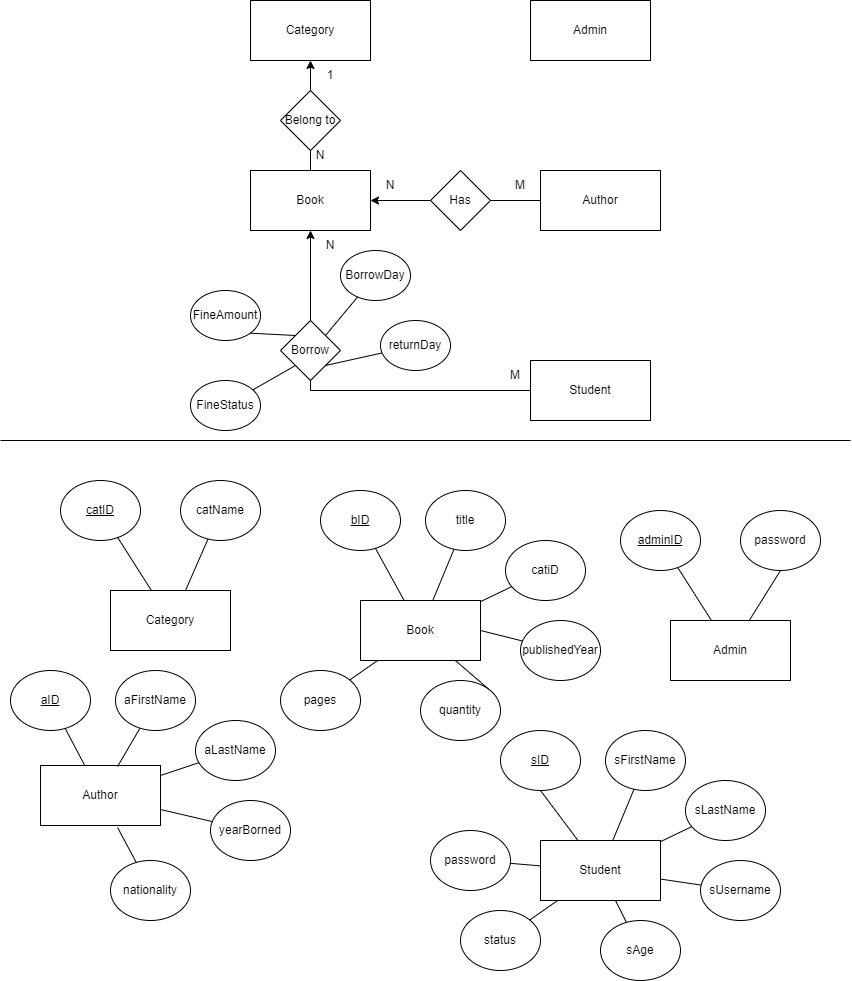
### Adding constraint (primary key & foreign key)

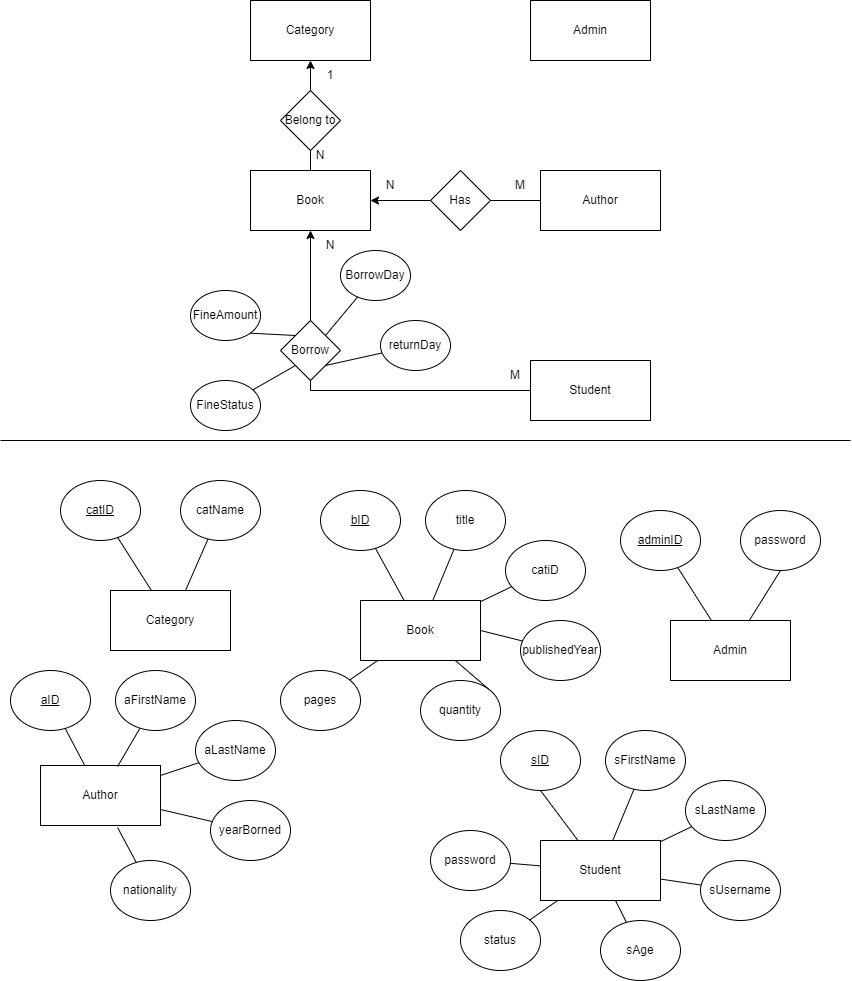


* With the primary keys, I put all of them in the IDs of each table with the odd one is the AuthorBook table where the key is put in both BookID and AuthorID since this is a linking table for 2 tables, that way the data is unique.
* As a mentioned, there are tables that have connection and that is set by foreign keys. These table are AuthorBook, Book, Borrow and Fine.
* As for AuthorBook, since it connects the 2 table Books and Author, there are 2 foreign keys to link with each. The bID of AuthorBook is connect with bID of Book and the same is for Author where aID is link with aID.
* With Book table, there are only one connect which is to Category. The catID is linked with catID.
* Similar to AuthorBook, Borrow links two table so there are two connections. These connections are to StudentID and BookID.

### Final ERD

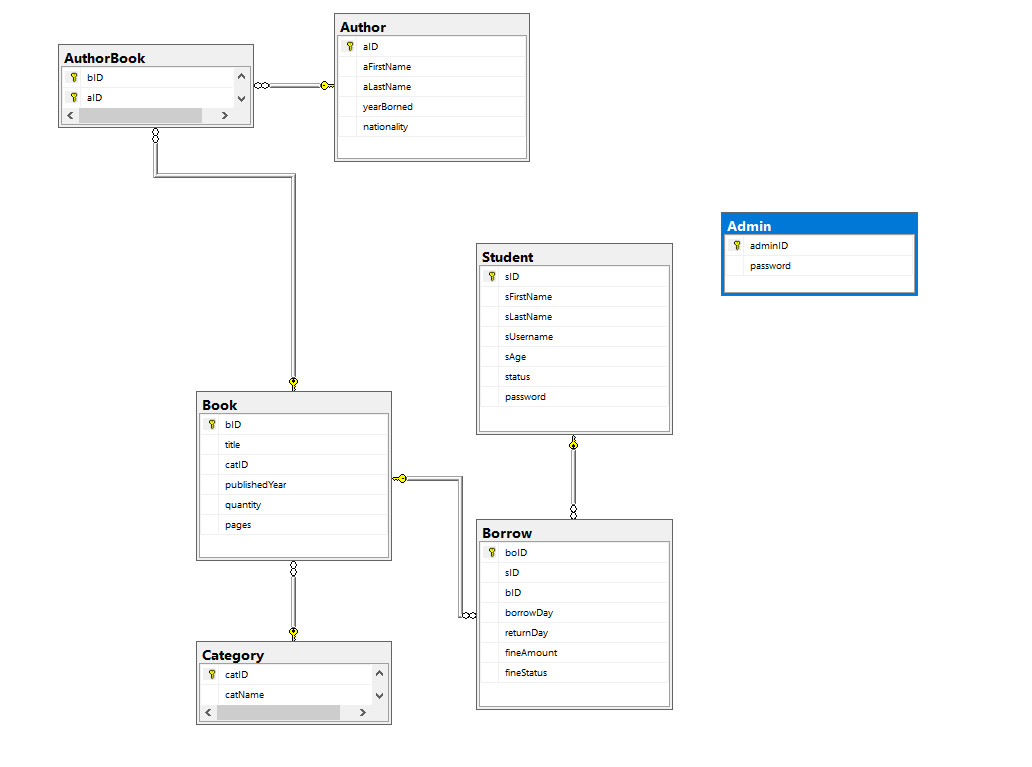
* Based on the given database, a diagram that illustrates the core structure of the system is needed. ERD or Entity Relationship Diagram will be displayed:





### Database diagram

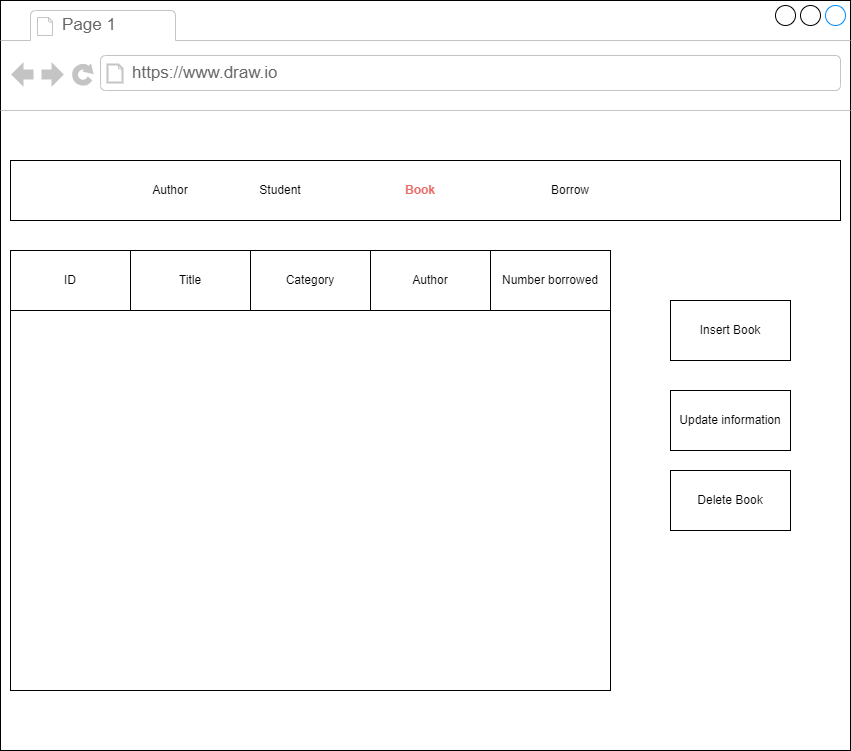
* The database diagram illustrates the structure of the database. This means information about the elements in one table, the primary keys and the links between tables. Here is the database diagram of the system:



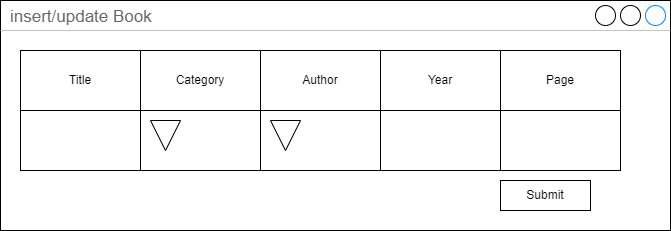
### Mock-up

* For the mock-up, I decided there are 4 main interfaces and small windows to perform any functions within. I will further explain the website functions in later section.

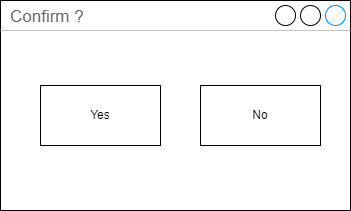
#### Book



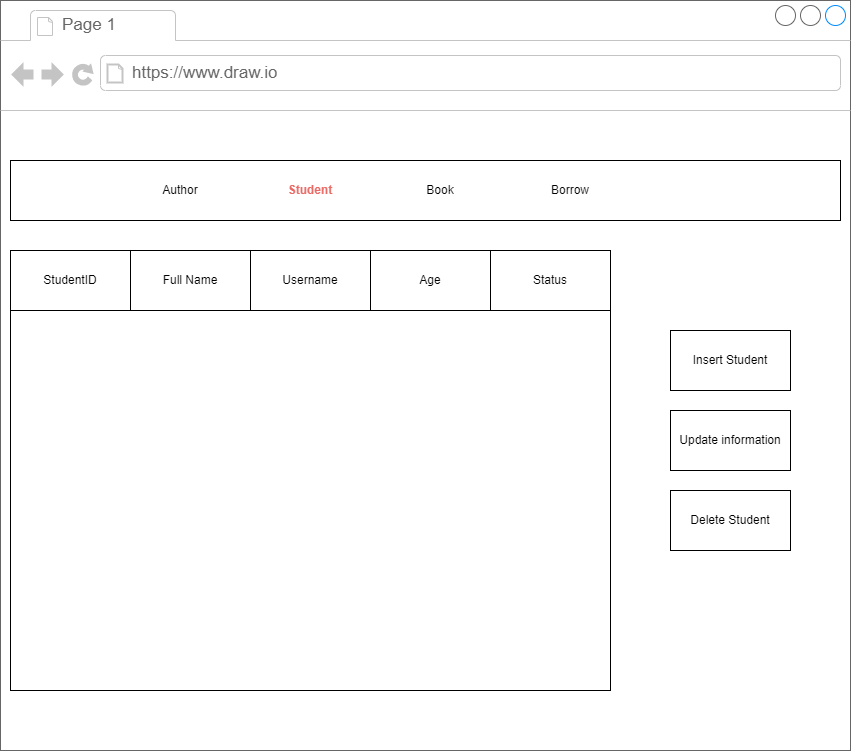
* This is the interface for the display of book as there is a main table of displaying information for books that are in the database. These are ID, Title, Category, Author and number being borrowed.
* There are 3 main functions here: Insert book, update information and delete book.
* The Insert function will deliver the user to a pop-up window where there is a table of information where they will fill up.



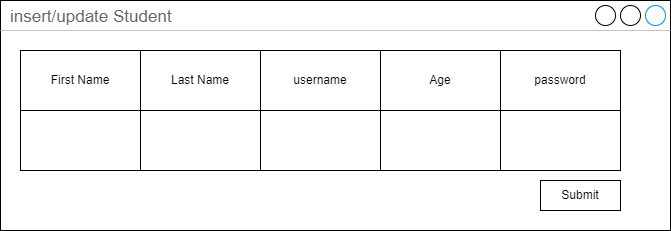
* Similarly, updating information will give us to the same window but the current information has been filled up. The user will change any information they want.
* Additionally, category and author are drop down option because those data already had a data and I only user to choose from that range.
* One important thing is that when inserting new book, the ID will be automatically added. This crucial for such unique data to not be changeable.
* Finally, deleting book will have another pop-up window of confirming. These will also delete the information in the borrowing and the linking with author table.



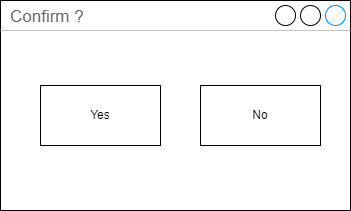
#### Student



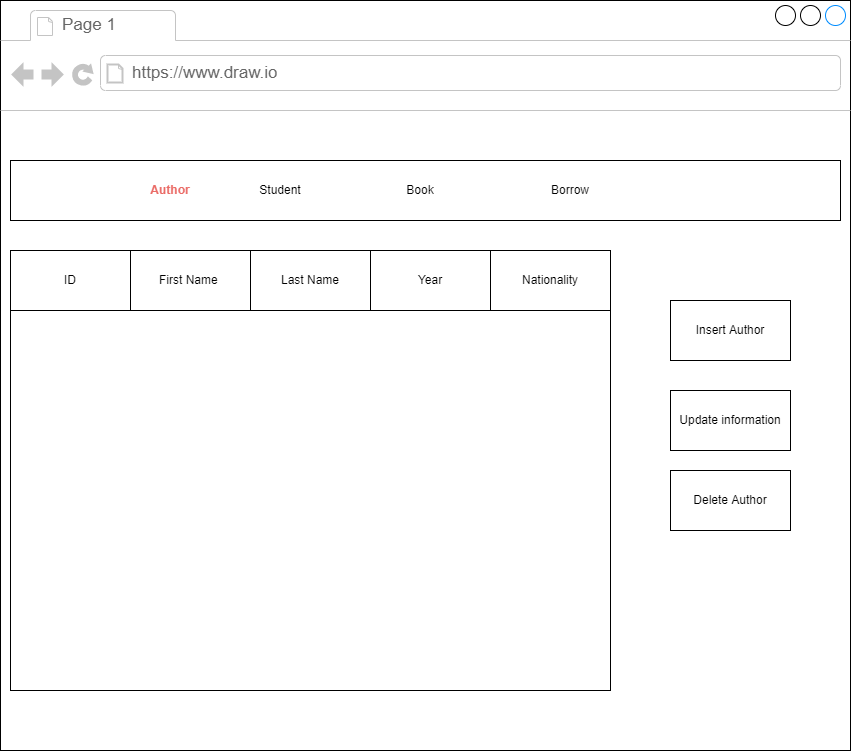
* Similar to Book, Student site will also display information of the student that are registered in the system. These information are their ID, Name, username, Age and status.
* There are 3 functions like Book: inserting new student, update information of exists student and delete a student record.



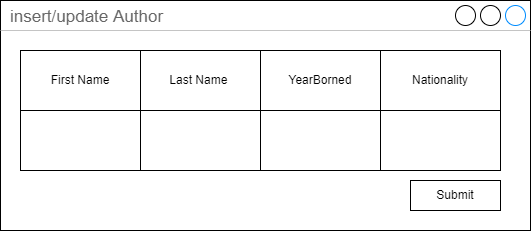
* This is the pop-up window of the insert and update just like previous mentioned Book. Users need to input information like first name, last name, username, age and password.
* Updating will have the same window with the information fill up so the users only need to make changes.
* Deleting will have a confirmation window. When deleting, the data in the Student as well as their borrow record will be eliminated.



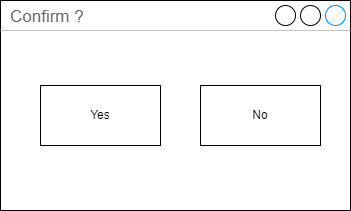
#### Author



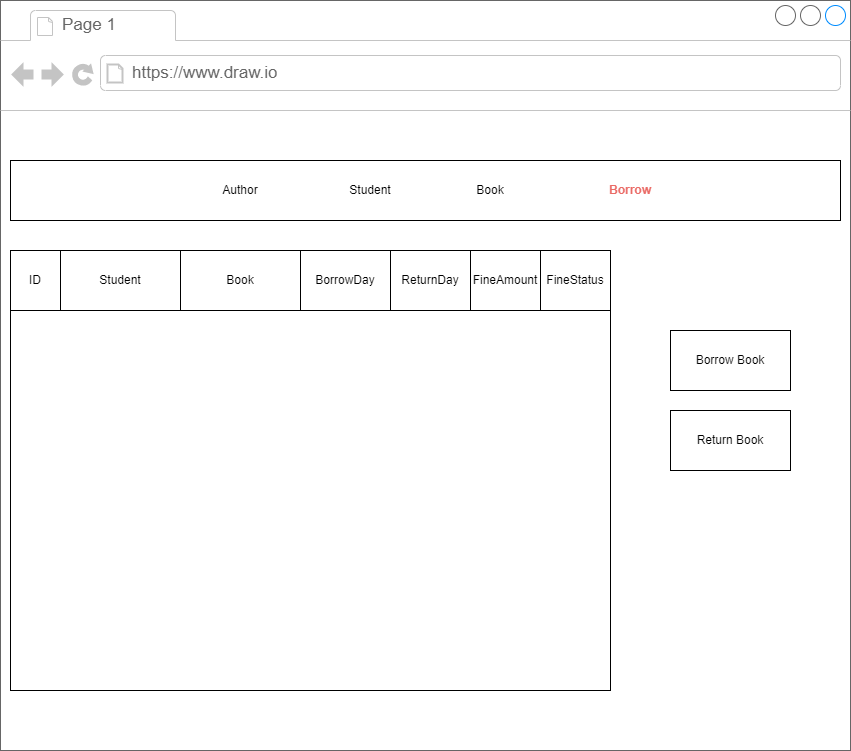
* As for the author site, the same can be said about this. There is a table display information of the author which are their ID, first name, last name, their year of birth and their nationality.
* There are also 3 main functions of this just like the other two: inserting, updating, deleting



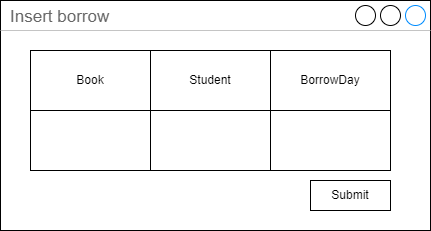
* The similar pop-up window will appear with information like first name, last name, birth year and nationality. This will be applied to both inserting and updating.
* Deleting will have a confirmation window.



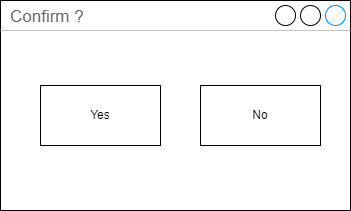
#### Borrow



* The same core structure is similar to other sites. There is table displaying ID of the borrow, the student that borrow, the book that was borrow, the day of taking and returning, the amount of fine and the status of that.



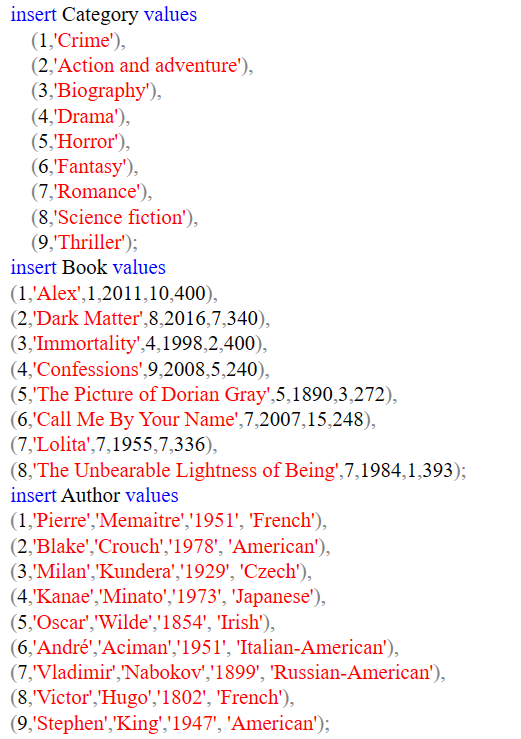
* The borrow window will have the book, the student borrows and the borrow day.
* As for the pay fine, only the confirmation window is presented. After that other work will be for computer to handle.

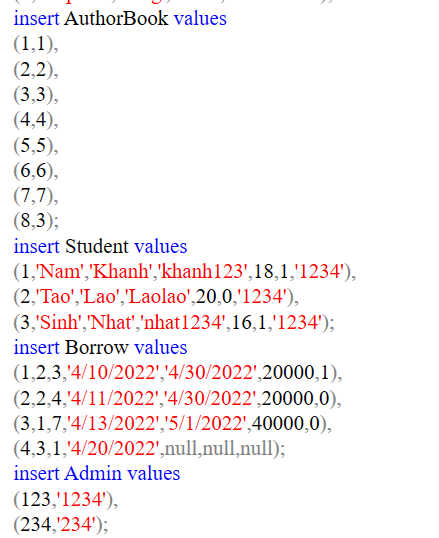


# Produce Queries

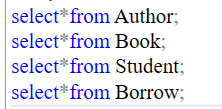
## Queries to INSERT data with illustrations

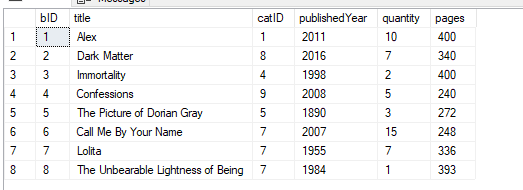
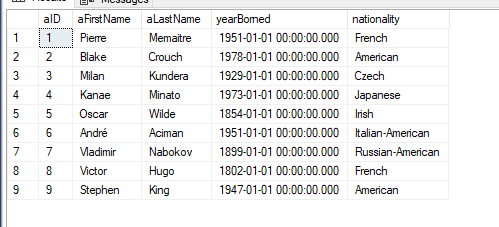
* The process inserting data to the table is important for the database to function smoothly.

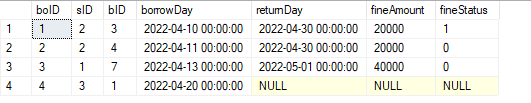
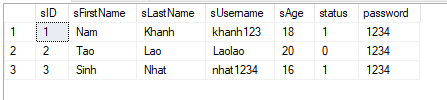




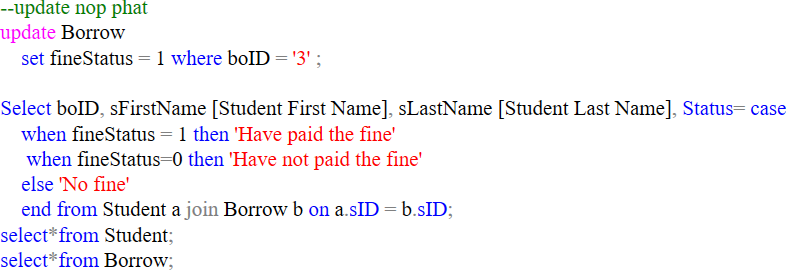
* I will provide 4 example of 4 main tables.



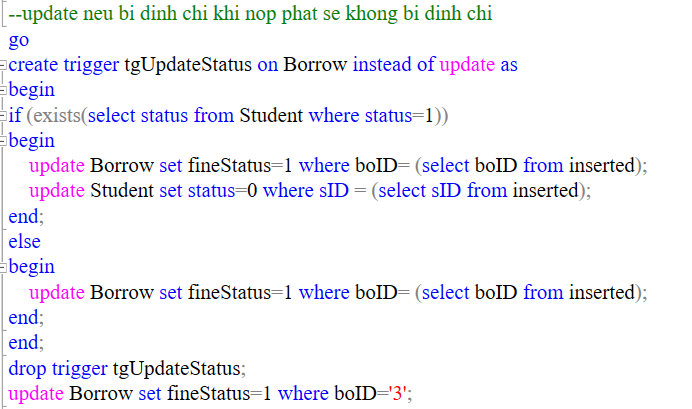




## Queries to UPDATE data



* This is the update for when a student pays the fine. Additionally, the status is set in the bit so there are 3 options: have paid, have not paid and no need to pay.



* However, I made a trigger so that if the student got suspended for not paying the fine but then afterward pay the fine. The system will change the fine to ‘have pay’ and the status of them to ‘active’ instead of ‘suspended’ like before.
* If the pay fine student is not suspended then the system will only change the fine to paid.
* For example, the following result will clear out this function. The first result is the beginning and we can see the student 1 and 3 has been suspended.

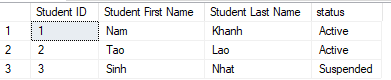


* In the borrow table we can see student 1 have not paid the fine.



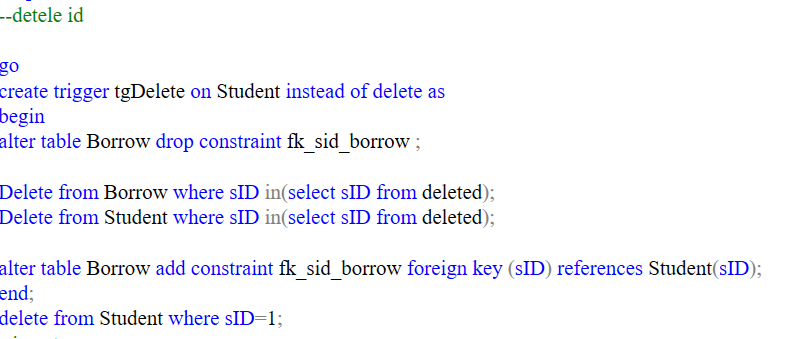
* So, when he pays the fine the system will work out different result.



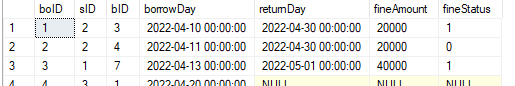
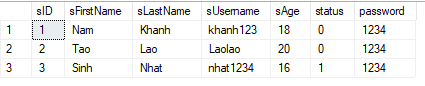




## Queries to DELETE data



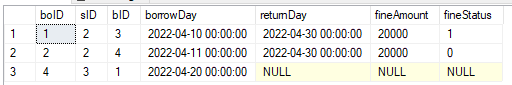
* In this deleting situation, I will give an example when deleting a student. It is appropriate to delete all of the record of the student. So, in this case I will use trigger instead of delete to execute multiple codes.
* Since we are deleting a data that link to other tables which is not allowed in sql ssms, we need to delete the link and add them back afterward.
* Beside from deleting the record of student in the student table, we also need to delete the record of them borrowing book otherwise the data will not make sense.
* As we can see from the example, the student 1 is exists in both student and borrow table.



* When I delete the record should be wiped out in both tables.

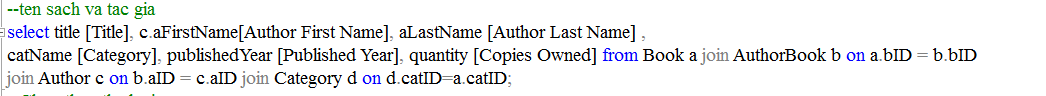






## Queries to SELECT data

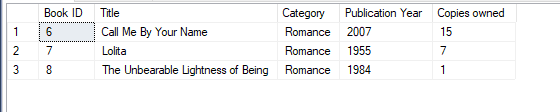
* In the last assignment there are many situations when checking the data with different data types. I will take the following section directly from my assignment 1.
* I will review some cases for many situations that the database might come to practice. I will provide the code and the result.
* Select books and their authors:





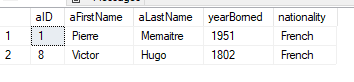
* Select books based on a category. In this case I choose romance:





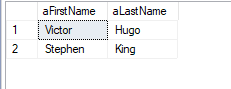
* Select authors based on their nationality. In this case I choose the nationality to be French:





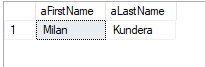
* Select authors where there are no records of their work in the system:



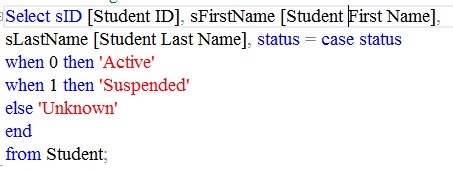


* Select authors that have more than one book in the library:



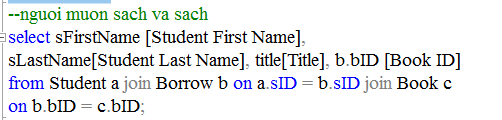


* Check the status of the Student:



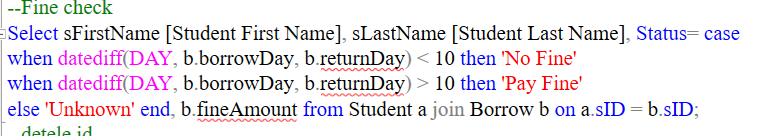


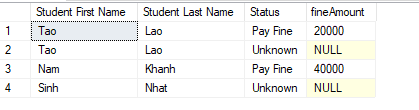
* Select the students and the books they borrowed:



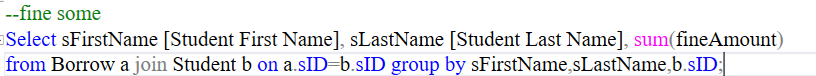


* Check the fine of the student:





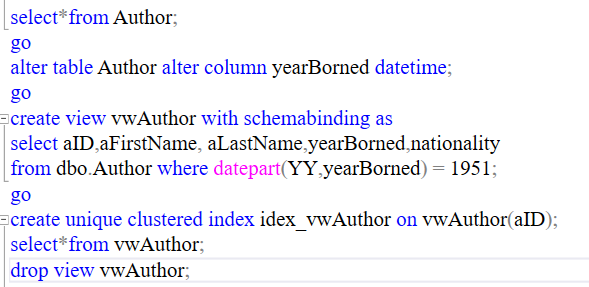
* In this case I had the duration of borrowing to be 10 days, if exceed the fine will be accounted.
* View the summary of each student fine.





## Advanced queries + Evaluation on effectiveness of the systems

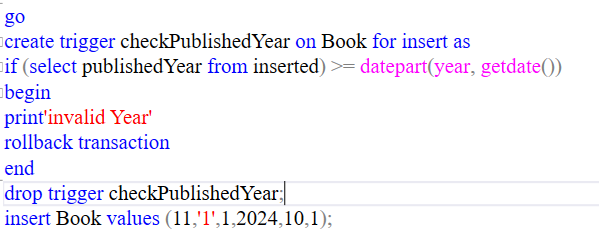
### Create view and index



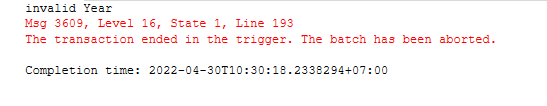
* In this I created a view from the author table of their birth year to be 1951.



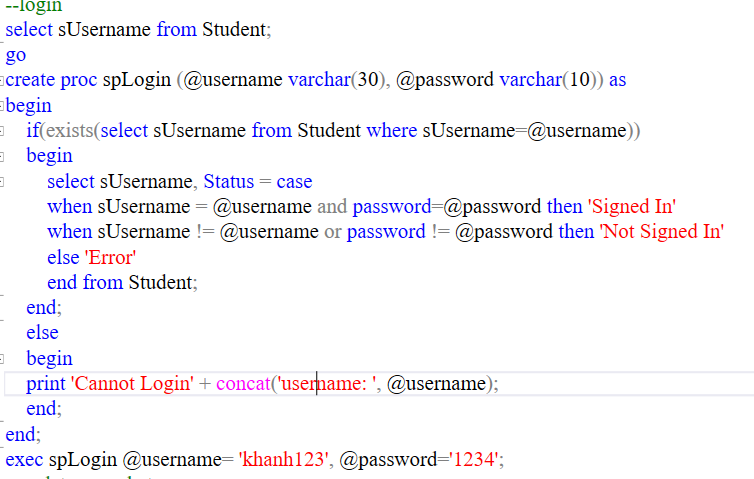
### Insert trigger



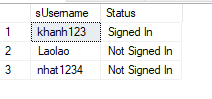
* For this I created a check trigger for insert. When a new book is insert, the year of it must be smaller than the current year. In this example, I had the insert year to be 2024 which is expected to be unallowed.



### Login procedure



* For an example of login, I will use Student for login. As for such I will have to create two new variables call @username and @password to compare with that of the database.
* The login only executes if there is existed a username similar to the input otherwise the system will print cannot login to that.
* If both the username and password are identical then the user will be logged in. If however either the username or password match then they will not logged in.
* The result:



### Evaluation on effectiveness of the systems

* For the effectiveness of the design of the system, all of the required CRUD are existed. Create is presented when making the table. Read is for the select and check data in the previous section. Update is presented when updating the status of the student and the fine. An example of Delete is also existed when deleting a student.
* A disadvantage of the system is that it is yet to connect with any website or program since the development happened purely in the database program. As a result, any practical problems are yet to be spotted.
* Another drawback of the system is the simplicity in the variety of the data. This means there should include more additional information for student, book, author. This can be solved quickly since the main core of the data had been created.

# Test the system

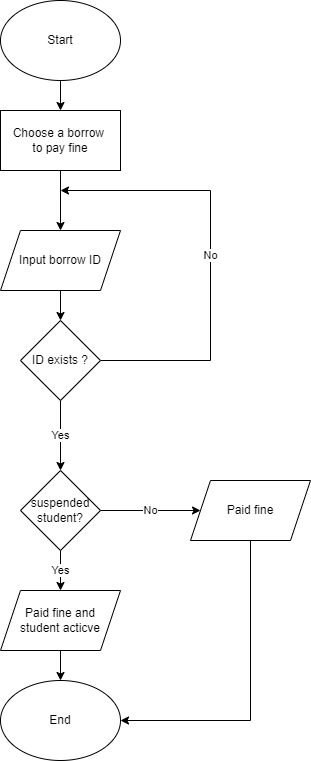
## Test cases

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Test | Expected result | Result |
| 1 | Inserting data | Insert successfully | Pass |
| 2 | Check data inserted | Data display correctly | Pass |
| 3 | Deleting member | The member is deleted on Student and Borrow table | Pass |
| 4 | Update when student pay fine | Update the status to have paid fine  If suspended then change to active | Pass |
| 5 | Use trigger to check Book year | If the year inserted is larger than the current year then  Insert fail | Pass |
| 6 | Login | If username and password match then login  If username and password don’t match then not signed in  If the username not exists then cannot login | Pass |
| 7 | Delete all database | The database is deleted | Pass |

## Flowcharts of the system

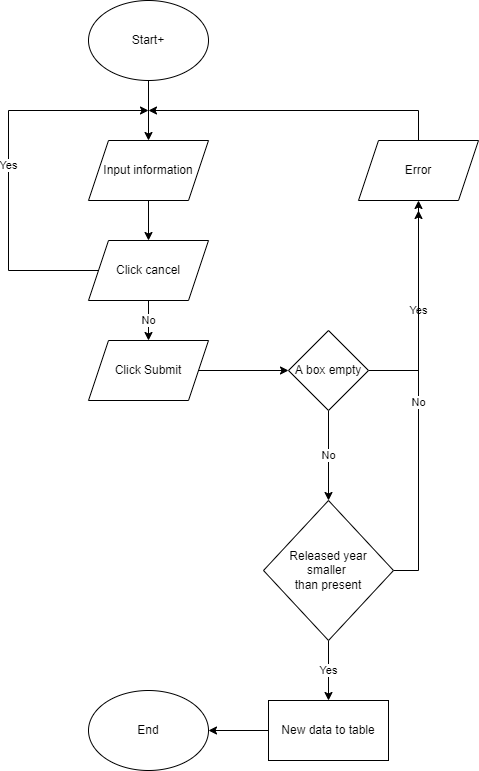
### Update (pay fine)

* For this example, I will update the fine status when a student pay the fine.



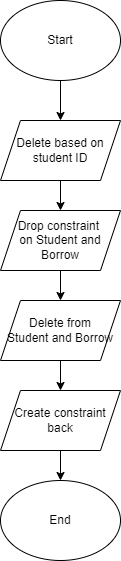
### Insert

* For creating I will take ‘Book’ as an example:



### Delete

* For deleting, I will delete a student record:



### Login

