A Project Report on :

“DNS LIBRARY DATABASE SYSTEM”

Prepared by :

School ID	Student Name
1076866	Shannon Mascarenhas
1059666	Deep Desai
1060679	Niket Sagar

Subject	:	CSCI-760 Database Systems
Professor	:	Dr. Huanying Gu
Semester	:	Fall 2016

Index

1. Introduction

We have developed desktop application using below tools and requirements.

Project Environment :

Eclipse Standard/SDK - Version: Kepler Service Release 2

Build id: 20140224-0627

Mysql workbench 6.2

Ubuntu 15.04

mysql-connector-java-5.1.40-bin

Java Version : 1.7 JDK

Library Management System is one most widely used in universities and schools for book management and used for public library to get the analytics of readers and their reading habits.

We have developed desktop application with aim to fulfilled the project requirement and learing database management system though design, developed and implement real time system to manage the books.

Our DNS Library has multiple branches in the different areas and neighbourhoods of New York City.

2. ER data model design

2.1 List of entities and their attributes :

This database design has 9 Entities. Which are book, branches, reader_details, total_no_of_books, reserve, borrow_return, book_details, authors, publishers. Belowe is a brief explanation for the tables and what they contain.

- Table book :

  This table contains books with different ISBNs and their multiple copies. It has two attributes.

  Attributes :

  book_id (int(11)unsigned),

  ISBN (int(11)unsigned) .

- Table branches :

  This table contains the information regarding all the branches of our library. It has three attributes.

  Attributes :

  library_id (int(11)unsigned) ,

  library_name(varchar(30)) ,

  library_location(varchar(50))

- Table readers_details :

  This table contains the information regarding all the readers that are registered in the library. It has four attributes.

  Attributes :

  reader_id (int(11)unsigned),

  reader_name (varchar(20)),

  reader_address (varchar(50)),

  phone_number (int (10)unsigned)

- Table book _details :

  This table has all the information about all the books. It has three attributes.

Attributes :

ISBN (int(11)unsigned),

published_date (date),

title (varchar(50))

- Table authors :

  This table contains information of the authors whose books are available in this library. It has two attributes.

  Attributes :

  author_id (int(11)unsigned),

  author_name (varchar(20))

- Table publishers :

  This table contains information about all the publishers whose books are available in this library. It has three attributes.

  Attributes :

  publisher_id (int(11)unsigned),

  publisher_name (varchar(20)),

  publisher_address (varchar(50))

2.2 Relationships and their attributes :

- Table borrow :

  This table contains all the entries of the books reserved by all the readers from any branches. This table has five attributes.

  Attributes :

  borrow_id (bigint(20)unsigned)

  borrow_datetime (datetime),

  due_date (date), (derived)

  return_datetime (datetime),

  fine_paid_reader (float)

- Table reserve :

  This table contains all the entries of the books reserved by all the readers from any branches. This table has three attributes.

  Attributes :

  reserve_id (bigint(20)unsigned),

  reserve_datetime (datetime),

  status (varchar(20))

- Table total_no_of_copies :

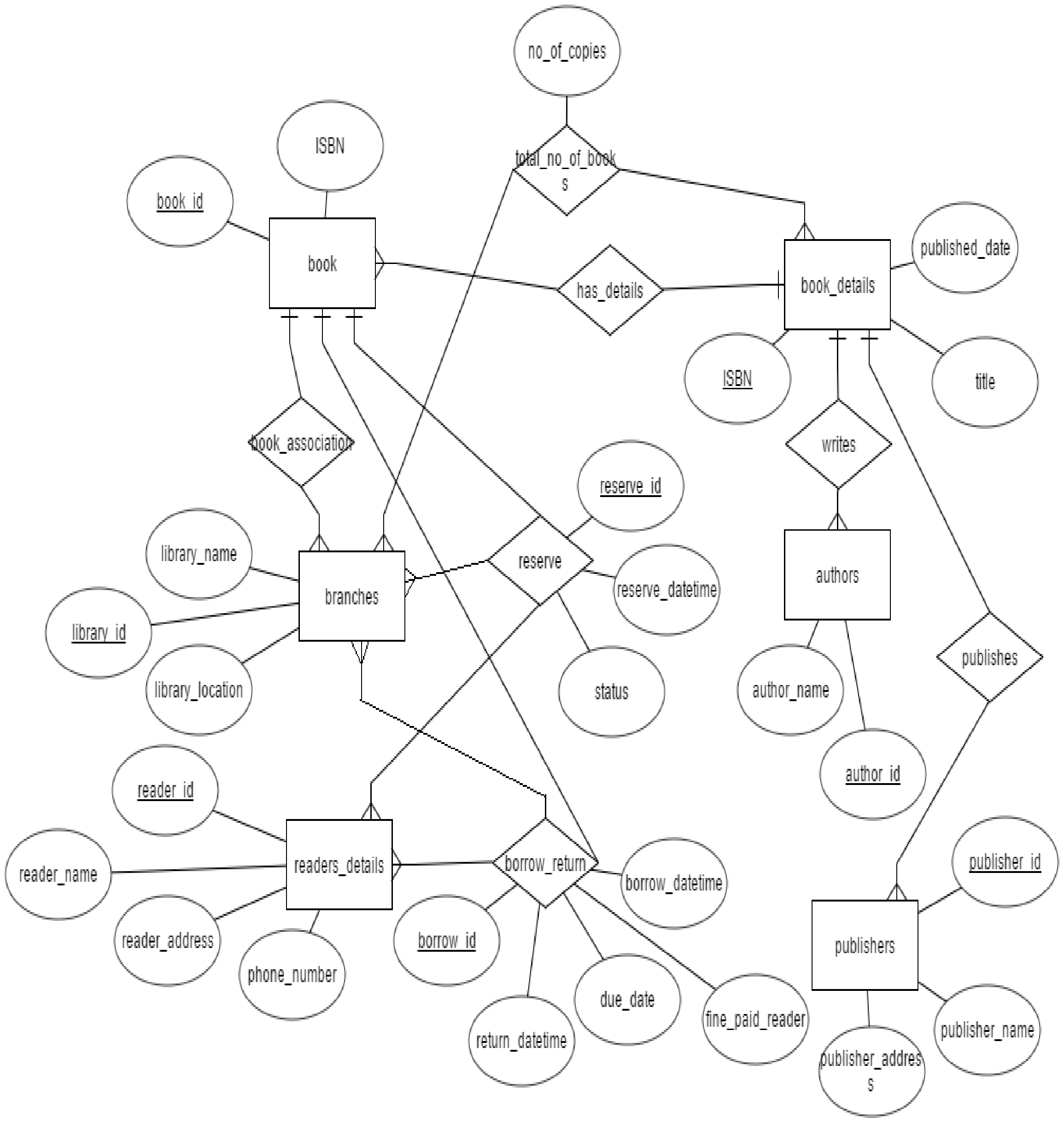
  This table contains the information regarding total number of copies of each individual book. It has one attribute.

  Attributes :

  no_of_copies (int(5)unsigned) (derived)

- Relationship book_association
- Relationship has_details
- Relationship writes
- Relationship publishes

2.3 ER diagram to our library database system :



3. Logical Design of the Database

3.1 Mapping of relations from ER diagram :

book (book_id, ISBN)

branches (library_id, library_name, library_location)

readers_details (reader_id, reader_name, reader_address, phone_number)

book_details (ISBN, title, published_date)

authors (author_id, author_name)

publishers (publisher_id, publisher_name, publisher_address)

book_association (book_id, library_id)

total_no_of_books (ISBN, library_id, no_of_copies)

has_details (book_id, ISBN)

reserve (reserve_id, reserve_datetime, status, reader_id, library_id,book_id)

borrow_return (borrow_id, borrow_datetime, due_date, return_datetime, fine_paid_reader)

writes(ISBN, author_id)

publishes (ISBN, publisher_id)

3.2 Entity and Referential integrity constraints :

book : referential integrity constraint

book_association : referential integrity constraint

book_details : referential integrity constraint

total_no_of_books : referential integrity constraint

branches : referential integrity constraint

borrow_return : referential integrity constraint

reserve : referential integrity constrain

4. Relational Database Design

4.1 1NF :

For a database to be in 1NF, it should have no composite or multivalued attributes or nested relations. In the database that we have created, there is no such scenario. That is why it is in 1NF.

4.2 2NF :

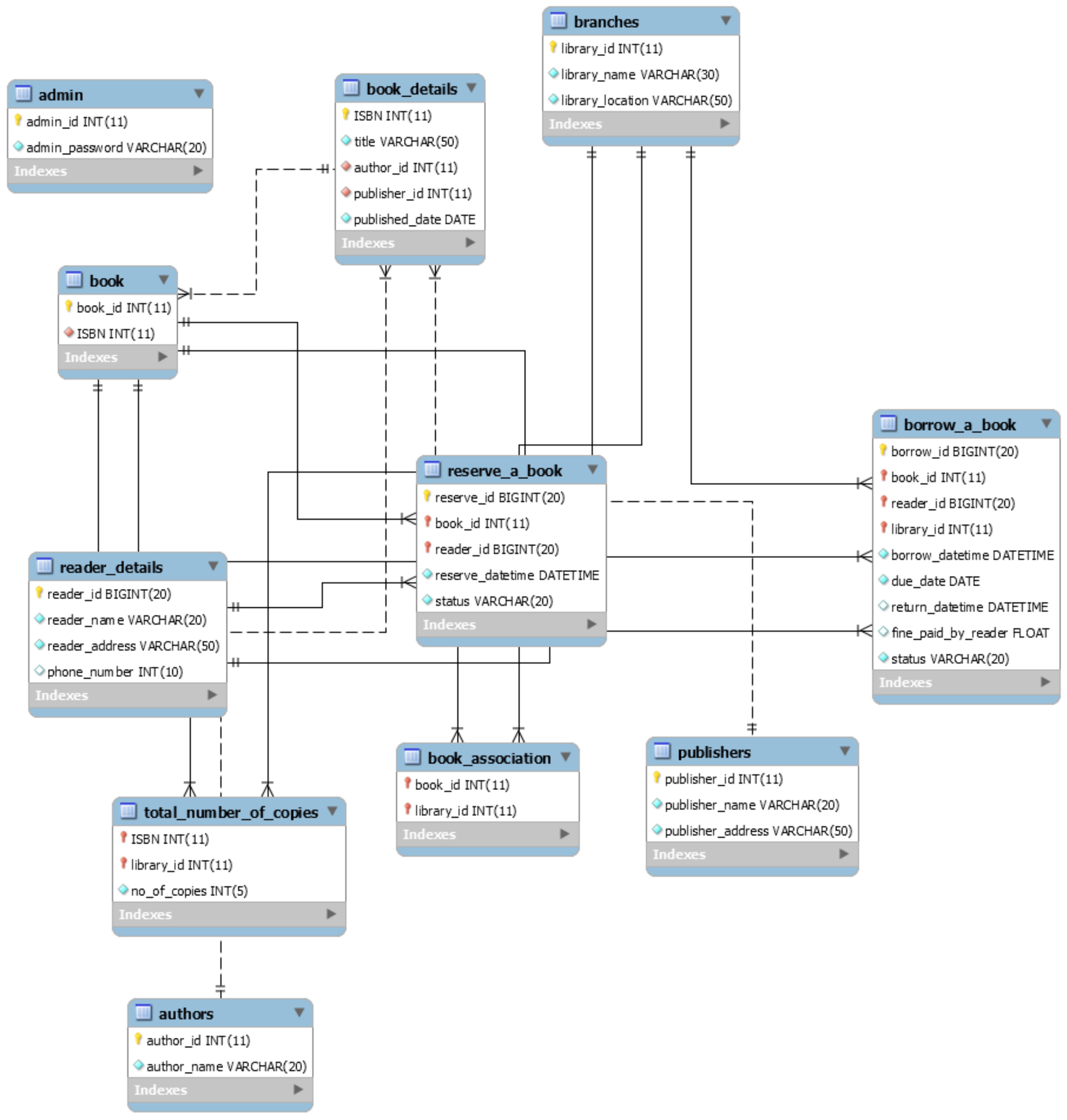
For a database to be in 2NF, all non prime attributes should fully be functionally dependent on primary key or candidate key. Which is applicable to our database too. That is why we can say, the database is in 2NF.

4.3 3NF :

For a database to be in 3NF, there should not be any transitive dependency. No attribute should be transitively dependent on the primary key. This condition is also satisfied in this database. So the database is in 3NF.

5. Implementation of Databse and SQL Query

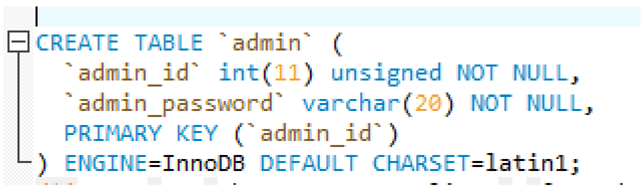
5.1 Database Schema of Library Management:



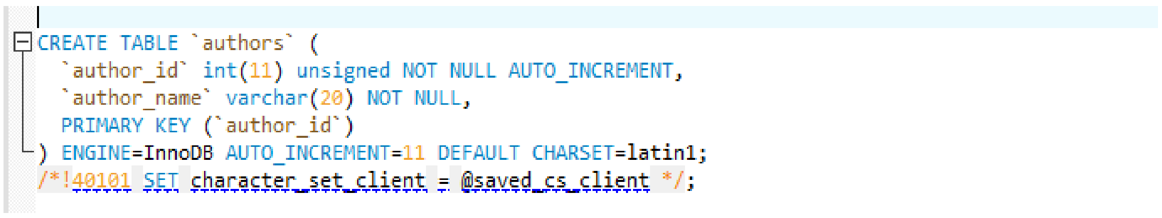
Here, we can the Database Schema which show the relationships with each table, Foreign Keys and One to One, One to Many, Many to One and One to One relationships. To create the database, we need to define the relationships, Foreign Keys and Datatype helps lot to get better system.

After creating the database schema, we need to create tables in the database systems which helps to store the data and using table we can get the required data.

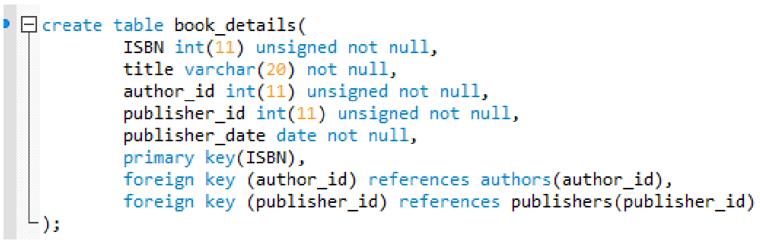
1. **Create admin tables:**


```
CREATE TABLE `admin` (
  `admin_id` int(11) unsigned NOT NULL,
  `admin_password` varchar(20) NOT NULL,
  PRIMARY KEY (`admin_id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

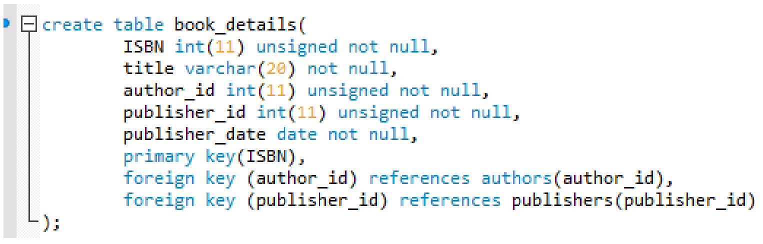
2. **Create author tables:**


```
CREATE TABLE `authors` (
  `author_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `author_name` varchar(20) NOT NULL,
  PRIMARY KEY (`author_id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

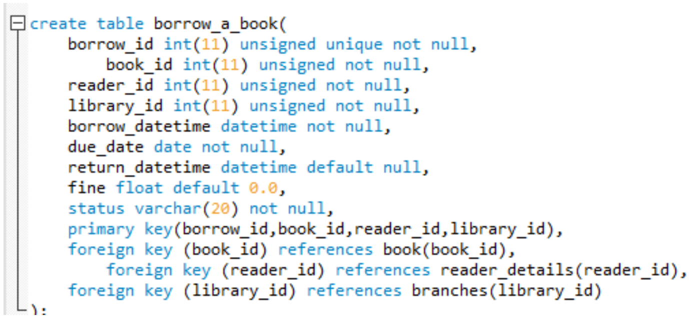
3. **Create book details tables:**


```
create table book_details(
        ISBN int(11) unsigned not null,
        title varchar(20) not null,
        author_id int(11) unsigned not null,
        publisher_id int(11) unsigned not null,
        publisher_date date not null,
        primary key(ISBN),
        foreign key (author_id) references authors(author_id),
        foreign key (publisher_id) references publishers(publisher_id)
);
```

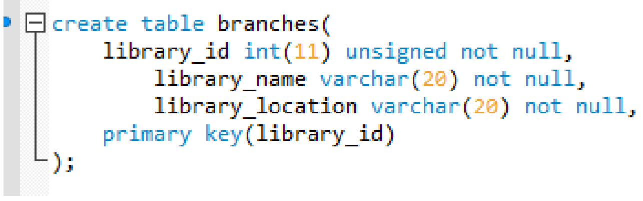
4. **Create book association tables:**


```
create table book_details(
        ISBN int(11) unsigned not null,
        title varchar(20) not null,
        author_id int(11) unsigned not null,
        publisher_id int(11) unsigned not null,
        publisher_date date not null,
        primary key(ISBN),
        foreign key (author_id) references authors(author_id),
        foreign key (publisher_id) references publishers(publisher_id)
);
```

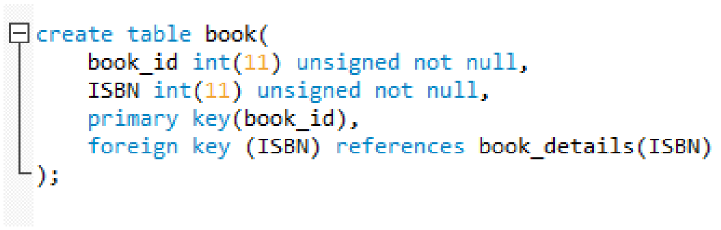
5. **Create borrow a book details**


```
create table borrow_a_book(
    borrow_id int(11) unsigned unique not null,
        book_id int(11) unsigned not null,
    reader_id int(11) unsigned not null,
    library_id int(11) unsigned not null,
    borrow_datetime datetime not null,
    due_date date not null,
    return_datetime datetime default null,
    fine float default 0.0,
    status varchar(20) not null,
    primary key(borrow_id,book_id,reader_id,library_id),
    foreign key (book_id) references book(book_id),
        foreign key (reader_id) references reader_details(reader_id),
    foreign key (library_id) references branches(library_id)
);
```

6. **Create branch table**

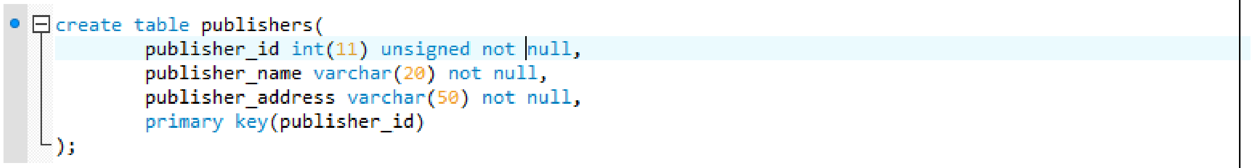

```
create table branches(
    library_id int(11) unsigned not null,
        library_name varchar(20) not null,
        library_location varchar(20) not null,
    primary key(library_id)
);
```

7. **Create author table**



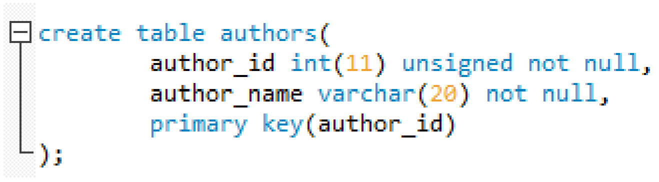
```
create table book(
    book_id int(11) unsigned not null,
    ISBN int(11) unsigned not null,
    primary key(book_id),
    foreign key (ISBN) references book_details(ISBN)
);
```

8. **Create Publisher table**



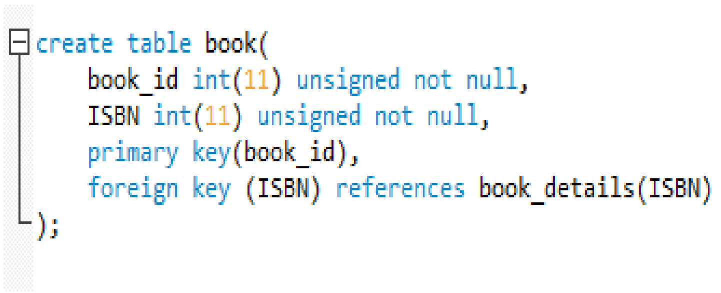
```
create table publishers(
        publisher_id int(11) unsigned not null,
        publisher_name varchar(20) not null,
        publisher_address varchar(50) not null,
        primary key(publisher_id)
);
```

9. **Create Author table**



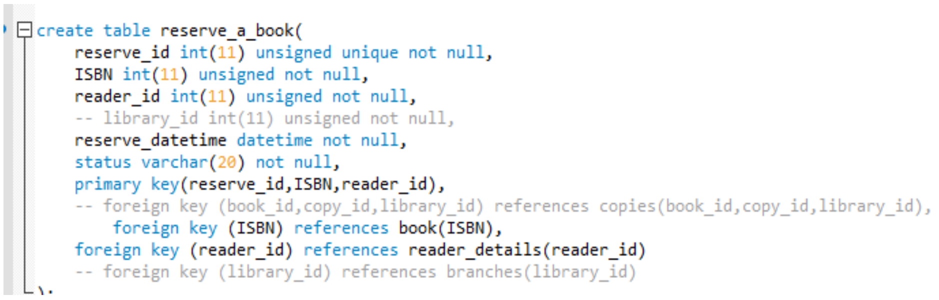
```
create table authors(
        author_id int(11) unsigned not null,
        author_name varchar(20) not null,
        primary key(author_id)
);
```

10. **Create Books table**



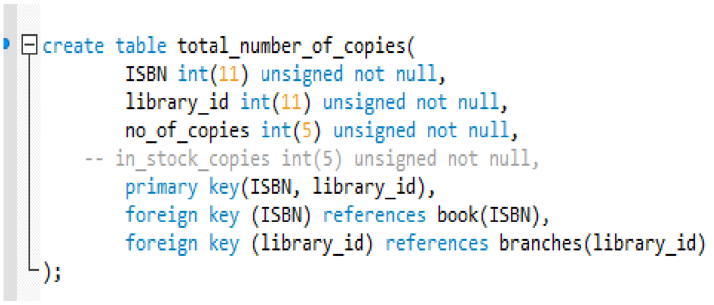
```
create table book(
    book_id int(11) unsigned not null,
    ISBN int(11) unsigned not null,
    primary key(book_id),
    foreign key (ISBN) references book_details(ISBN)
);
```

11. **Create Reserve a book table:**



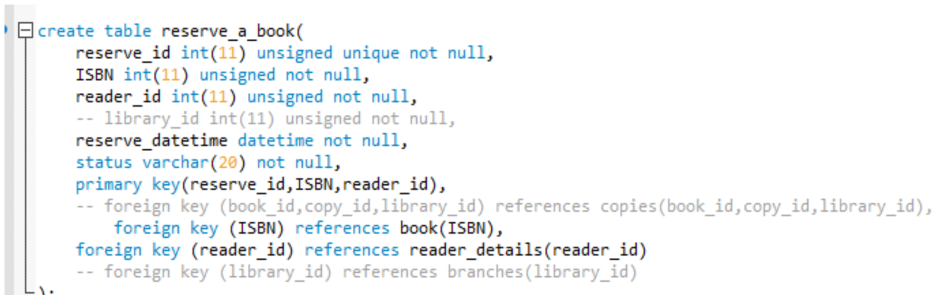
```
create table reserve_a_book(
    reserve_id int(11) unsigned unique not null,
    ISBN int(11) unsigned not null,
    reader_id int(11) unsigned not null,
    -- library_id int(11) unsigned not null,
    reserve_datetime datetime not null,
    status varchar(20) not null,
    primary key(reserve_id,ISBN,reader_id),
    -- foreign key (book_id,copy_id,library_id) references copies(book_id,copy_id,library_id),
        foreign key (ISBN) references book(ISBN),
    foreign key (reader_id) references reader_details(reader_id)
    -- foreign key (library_id) references branches(library_id)
);
```

12. **Create Total Number of Copies table:**



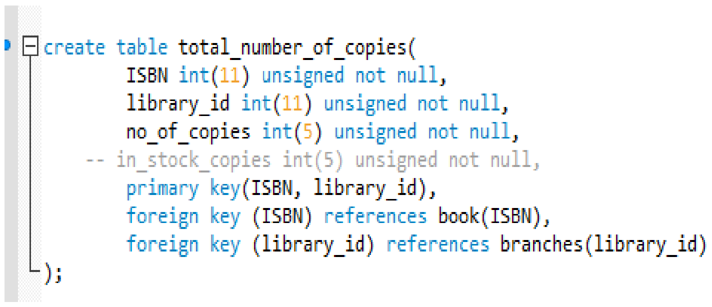
```
create table total_number_of_copies(
        ISBN int(11) unsigned not null,
        library_id int(11) unsigned not null,
        no_of_copies int(5) unsigned not null,
    -- in_stock_copies int(5) unsigned not null,
        primary key(ISBN, library_id),
        foreign key (ISBN) references book(ISBN),
        foreign key (library_id) references branches(library_id)
);
```

13. Create Reserve a book table:



```
create table reserve_a_book(
    reserve_id int(11) unsigned unique not null,
    ISBN int(11) unsigned not null,
    reader_id int(11) unsigned not null,
    -- library_id int(11) unsigned not null,
    reserve_datetime datetime not null,
    status varchar(20) not null,
    primary key(reserve_id,ISBN,reader_id),
    -- foreign key (book_id,copy_id,library_id) references copies(book_id,copy_id,library_id),
        foreign key (ISBN) references book(ISBN),
    foreign key (reader_id) references reader_details(reader_id)
    -- foreign key (library_id) references branches(library_id)
);
```

14. Create Total Number of Copies table:



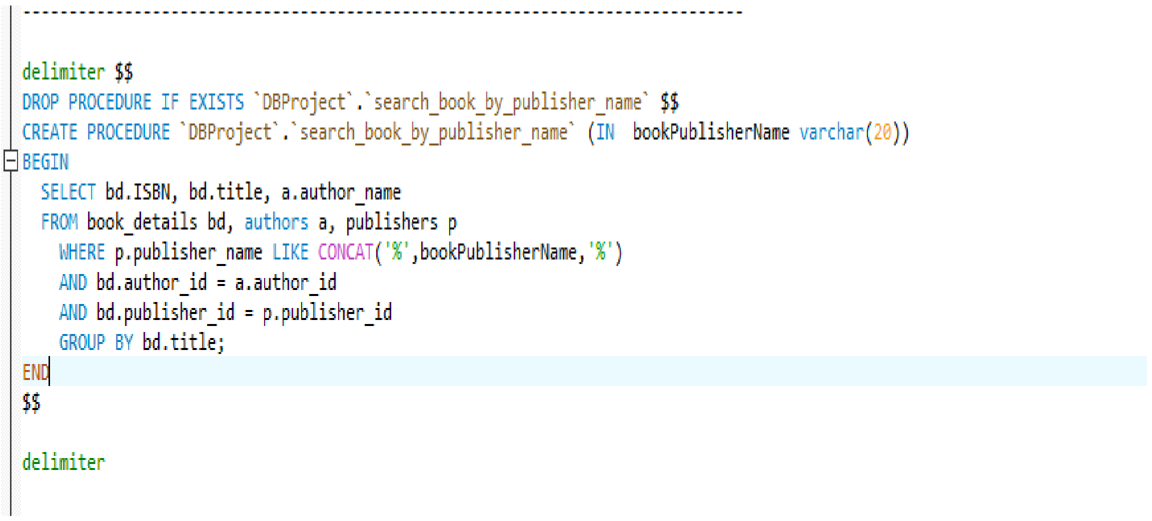
```
create table total_number_of_copies(
        ISBN int(11) unsigned not null,
        library_id int(11) unsigned not null,
        no_of_copies int(5) unsigned not null,
    -- in_stock_copies int(5) unsigned not null,
        primary key(ISBN, library_id),
        foreign key (ISBN) references book(ISBN),
        foreign key (library_id) references branches(library_id)
);
```

Provide the SQL statements that query the database:

Reader Functions Menu:

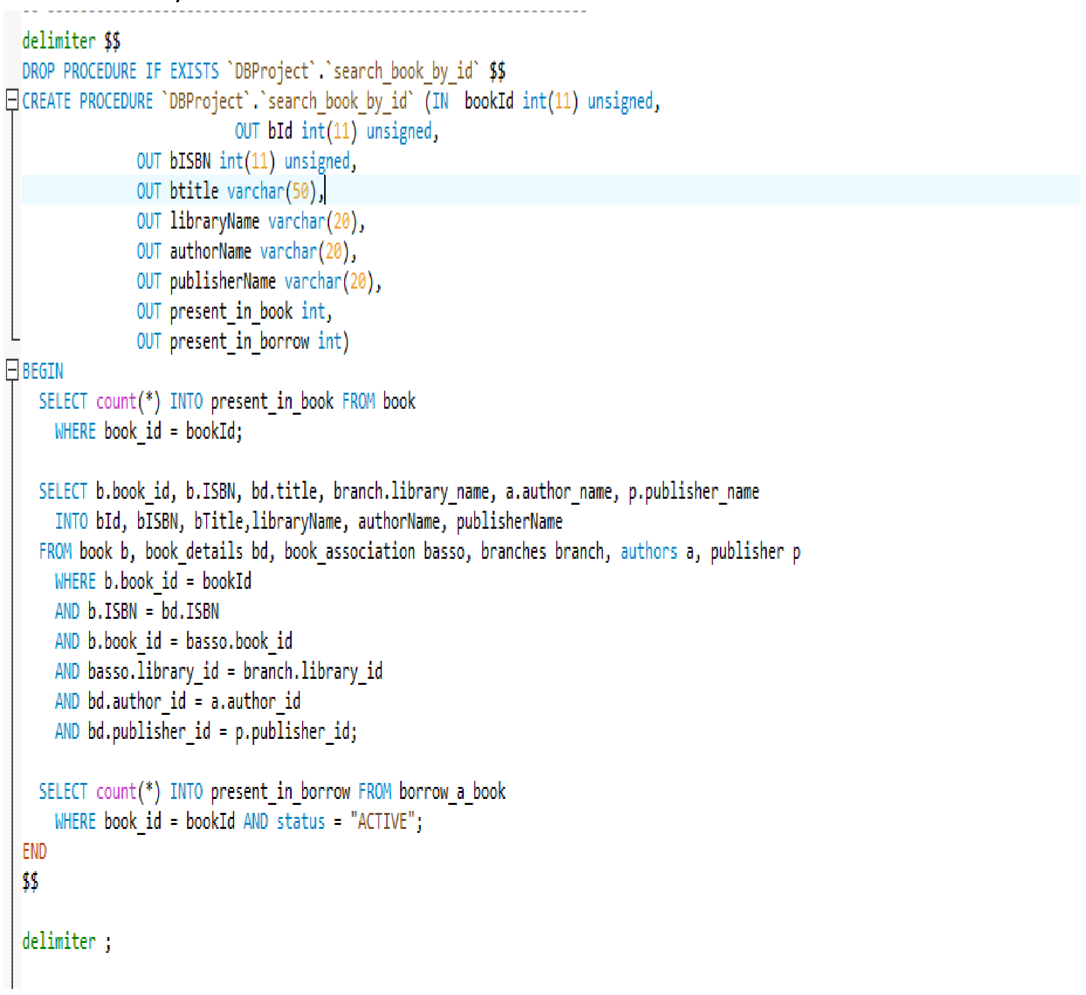
1. Search a book by ID, title, or publisher name

Search Book By Publisher Name :



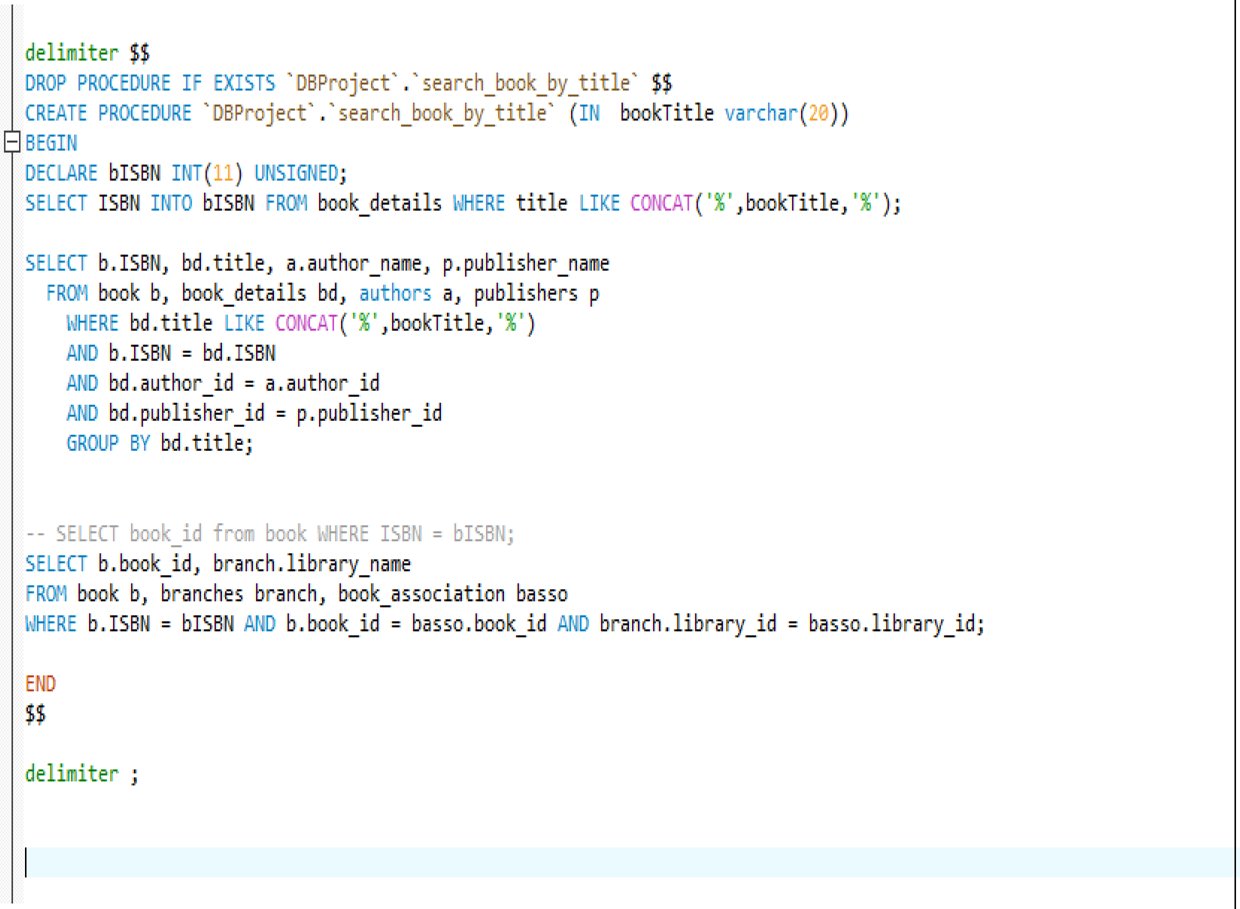
```
-----------------------------------------------------------------------------

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`search_book_by_publisher_name` $$
CREATE PROCEDURE `DBProject`.`search_book_by_publisher_name` (IN  bookPublisherName varchar(20))
BEGIN
  SELECT bd.ISBN, bd.title, a.author_name
  FROM book_details bd, authors a, publishers p
    WHERE p.publisher_name LIKE CONCAT('%',bookPublisherName,'%')
    AND bd.author_id = a.author_id
    AND bd.publisher_id = p.publisher_id
    GROUP BY bd.title;
END
$$

delimiter
```

Search book by ID:



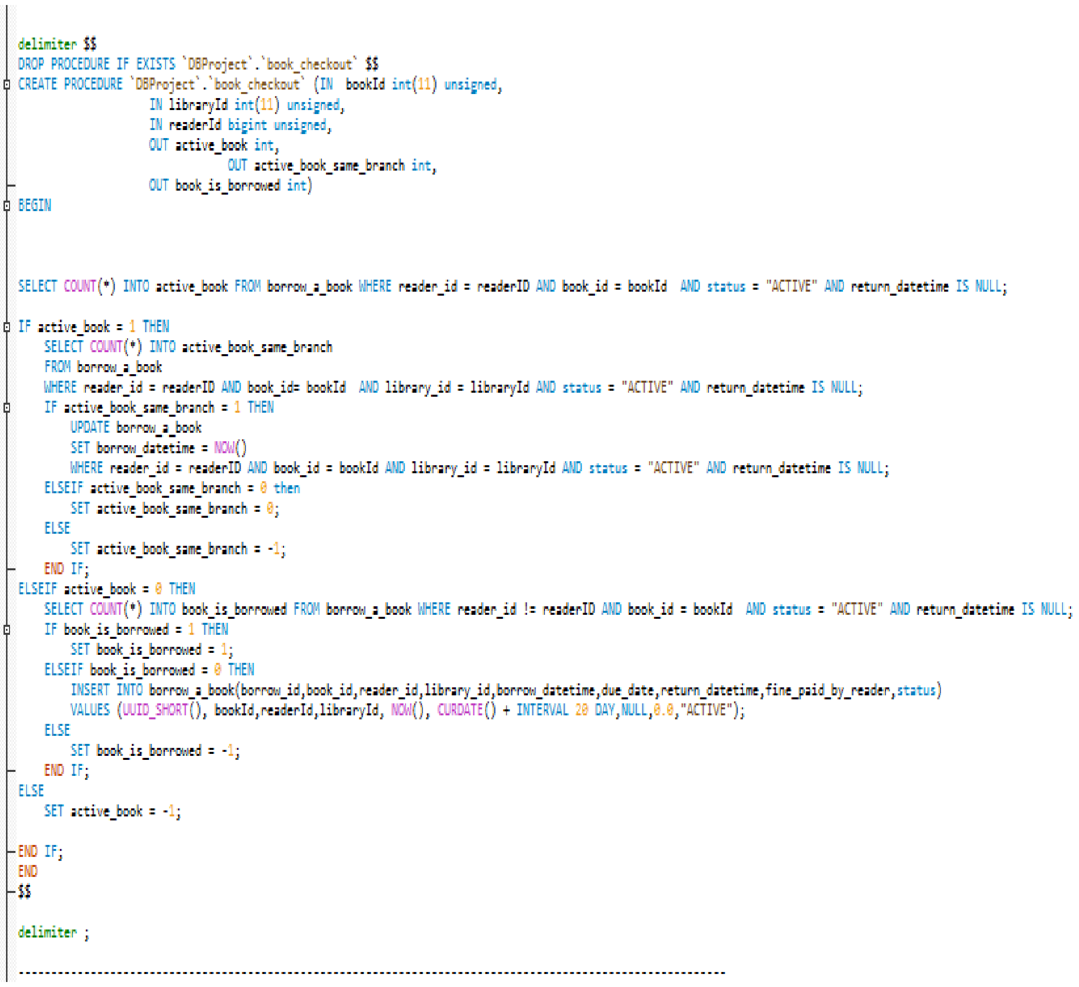
```
-- --------------------------------------------------------------------------------
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`search_book_by_id` $$
CREATE PROCEDURE `DBProject`.`search_book_by_id` (IN  bookId int(11) unsigned,
                          OUT bId int(11) unsigned,
              OUT bISBN int(11) unsigned,
              OUT btitle varchar(50),
              OUT libraryName varchar(20),
              OUT authorName varchar(20),
              OUT publisherName varchar(20),
              OUT present_in_book int,
              OUT present_in_borrow int)
BEGIN
  SELECT count(*) INTO present_in_book FROM book
    WHERE book_id = bookId;

  SELECT b.book_id, b.ISBN, bd.title, branch.library_name, a.author_name, p.publisher_name
    INTO bId, bISBN, bTitle,libraryName, authorName, publisherName
  FROM book b, book_details bd, book_association basso, branches branch, authors a, publisher p
    WHERE b.book_id = bookId
    AND b.ISBN = bd.ISBN
    AND b.book_id = basso.book_id
    AND basso.library_id = branch.library_id
    AND bd.author_id = a.author_id
    AND bd.publisher_id = p.publisher_id;

  SELECT count(*) INTO present_in_borrow FROM borrow_a_book
    WHERE book_id = bookId AND status = "ACTIVE";
END
$$

delimiter ;
```

Search Book By Title :



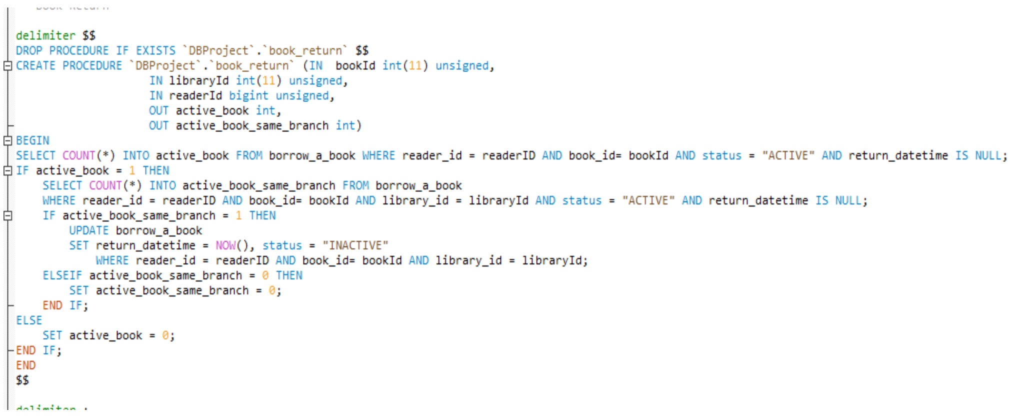
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`search_book_by_title` $$
CREATE PROCEDURE `DBProject`.`search_book_by_title` (IN  bookTitle varchar(20))
BEGIN
DECLARE bISBN INT(11) UNSIGNED;
SELECT ISBN INTO bISBN FROM book_details WHERE title LIKE CONCAT('%',bookTitle,'%');

SELECT b.ISBN, bd.title, a.author_name, p.publisher_name
  FROM book b, book_details bd, authors a, publishers p
    WHERE bd.title LIKE CONCAT('%',bookTitle,'%')
    AND b.ISBN = bd.ISBN
    AND bd.author_id = a.author_id
    AND bd.publisher_id = p.publisher_id
    GROUP BY bd.title;


-- SELECT book_id from book WHERE ISBN = bISBN;
SELECT b.book_id, branch.library_name
FROM book b, branches branch, book_association basso
WHERE b.ISBN = bISBN AND b.book_id = basso.book_id AND branch.library_id = basso.library_id;

END
$$

delimiter ;
```

2. Book checkout



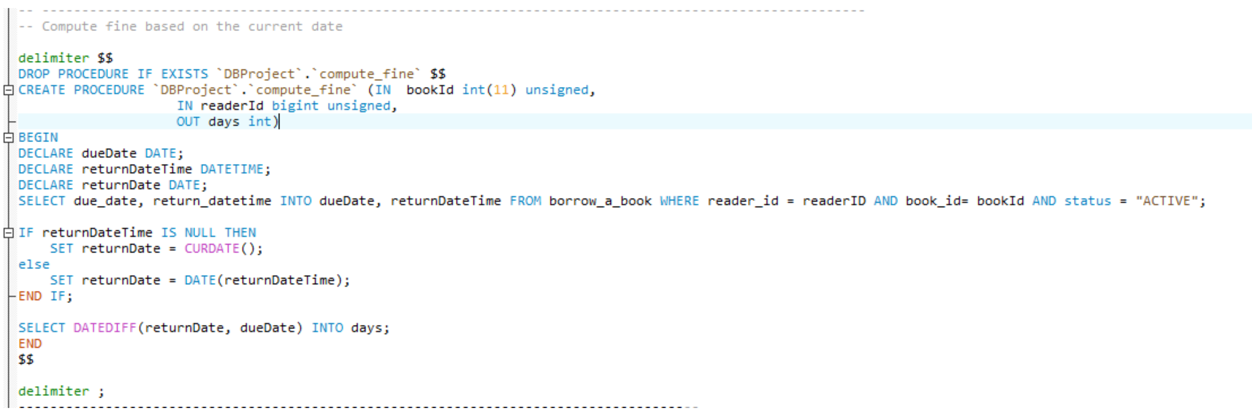
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`book_checkout` $$
CREATE PROCEDURE `DBProject`.`book_checkout` (IN  bookId int(11) unsigned,
                    IN libraryId int(11) unsigned,
                    IN readerId bigint unsigned,
                    OUT active_book int,
                                OUT active_book_same_branch int,
                    OUT book_is_borrowed int)
BEGIN



SELECT COUNT(*) INTO active_book FROM borrow_a_book WHERE reader_id = readerID AND book_id = bookId  AND status = "ACTIVE" AND return_datetime IS NULL;

IF active_book = 1 THEN
    SELECT COUNT(*) INTO active_book_same_branch
    FROM borrow_a_book
    WHERE reader_id = readerID AND book_id= bookId  AND library_id = libraryId AND status = "ACTIVE" AND return_datetime IS NULL;
    IF active_book_same_branch = 1 THEN
        UPDATE borrow_a_book
        SET borrow_datetime = NOW()
        WHERE reader_id = readerID AND book_id = bookId AND library_id = libraryId AND status = "ACTIVE" AND return_datetime IS NULL;
    ELSEIF active_book_same_branch = 0 then
        SET active_book_same_branch = 0;
    ELSE
        SET active_book_same_branch = -1;
    END IF;
ELSEIF active_book = 0 THEN
    SELECT COUNT(*) INTO book_is_borrowed FROM borrow_a_book WHERE reader_id != readerID AND book_id = bookId  AND status = "ACTIVE" AND return_datetime IS NULL;
    IF book_is_borrowed = 1 THEN
        SET book_is_borrowed = 1;
    ELSEIF book_is_borrowed = 0 THEN
        INSERT INTO borrow_a_book(borrow_id,book_id,reader_id,library_id,borrow_datetime,due_date,return_datetime,fine_paid_by_reader,status)
        VALUES (UUID_SHORT(), bookId,readerId,libraryId, NOW(), CURDATE() + INTERVAL 20 DAY,NULL,0.0,"ACTIVE");
    ELSE
        SET book_is_borrowed = -1;
    END IF;
ELSE
    SET active_book = -1;

END IF;
END
$$

delimiter ;

-----------------------------------------------------------------------------------------------------------
```

3. Book return



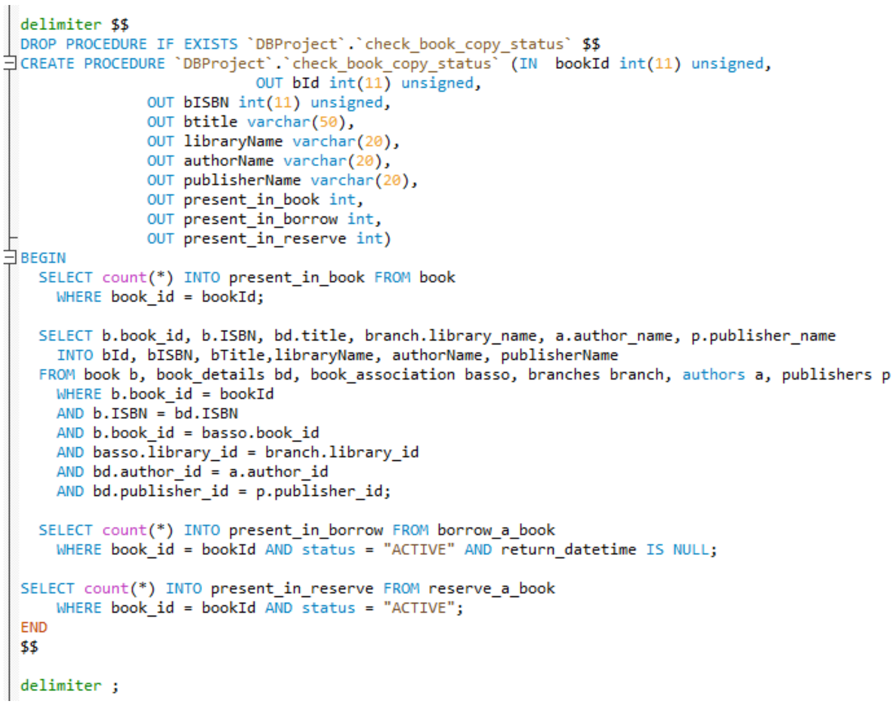
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`book_return` $$
CREATE PROCEDURE `DBProject`.`book_return` (IN  bookId int(11) unsigned,
                    IN libraryId int(11) unsigned,
                    IN readerId bigint unsigned,
                    OUT active_book int,
                    OUT active_book_same_branch int)
BEGIN
SELECT COUNT(*) INTO active_book FROM borrow_a_book WHERE reader_id = readerID AND book_id= bookId AND status = "ACTIVE" AND return_datetime IS NULL;
IF active_book = 1 THEN
    SELECT COUNT(*) INTO active_book_same_branch FROM borrow_a_book
    WHERE reader_id = readerID AND book_id= bookId AND library_id = libraryId AND status = "ACTIVE" AND return_datetime IS NULL;
    IF active_book_same_branch = 1 THEN
        UPDATE borrow_a_book
        SET return_datetime = NOW(), status = "INACTIVE"
            WHERE reader_id = readerID AND book_id= bookId AND library_id = libraryId;
    ELSEIF active_book_same_branch = 0 THEN
        SET active_book_same_branch = 0;
    END IF;
ELSE
    SET active_book = 0;
END IF;
END
$$
```

4. Compute fine for a book copy borrowed by a reader based on the current date.



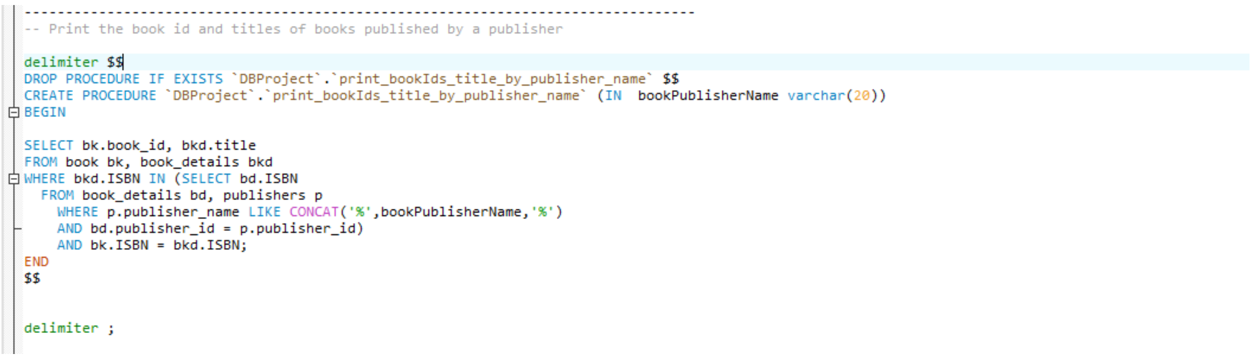
```sql
-- -----------------------------------------------------------------------------------------------
-- Compute fine based on the current date

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`compute_fine` $$
CREATE PROCEDURE `DBProject`.`compute_fine` (IN  bookId int(11) unsigned,
                    IN readerId bigint unsigned,
                    OUT days int)
BEGIN
DECLARE dueDate DATE;
DECLARE returnDateTime DATETIME;
DECLARE returnDate DATE;
SELECT due_date, return_datetime INTO dueDate, returnDateTime FROM borrow_a_book WHERE reader_id = readerID AND book_id= bookId AND status = "ACTIVE";

IF returnDateTime IS NULL THEN
    SET returnDate = CURDATE();
else
    SET returnDate = DATE(returnDateTime);
END IF;

SELECT DATEDIFF(returnDate, dueDate) INTO days;
END
$$

delimiter ;
-------------------------------------------------------------------------------------
```

5. Print the list of book reserved by a reader and their status.



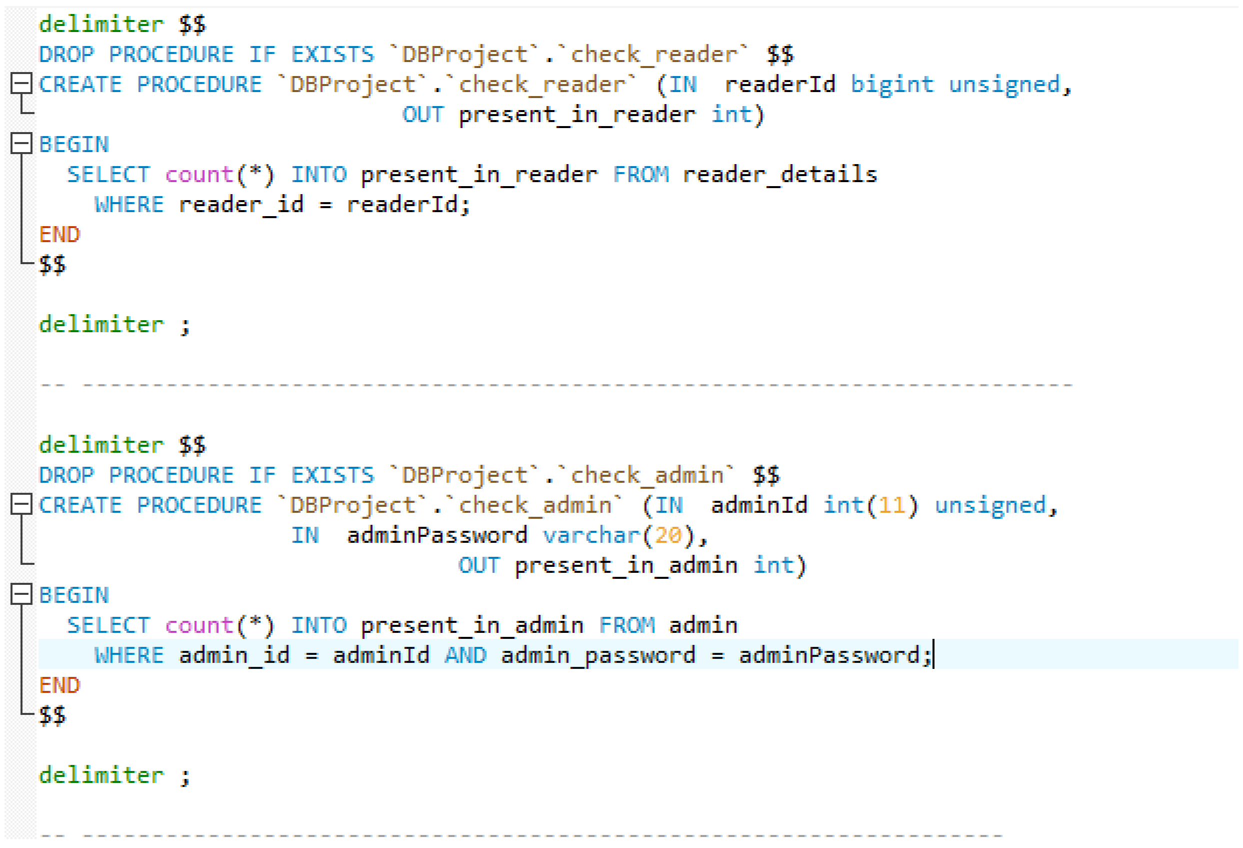
```sql
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`check_book_copy_status` $$
CREATE PROCEDURE `DBProject`.`check_book_copy_status` (IN  bookId int(11) unsigned,
                        OUT bId int(11) unsigned,
              OUT bISBN int(11) unsigned,
              OUT btitle varchar(50),
              OUT libraryName varchar(20),
              OUT authorName varchar(20),
              OUT publisherName varchar(20),
              OUT present_in_book int,
              OUT present_in_borrow int,
              OUT present_in_reserve int)
BEGIN
  SELECT count(*) INTO present_in_book FROM book
    WHERE book_id = bookId;

  SELECT b.book_id, b.ISBN, bd.title, branch.library_name, a.author_name, p.publisher_name
    INTO bId, bISBN, bTitle,libraryName, authorName, publisherName
  FROM book b, book_details bd, book_association basso, branches branch, authors a, publishers p
    WHERE b.book_id = bookId
    AND b.ISBN = bd.ISBN
    AND b.book_id = basso.book_id
    AND basso.library_id = branch.library_id
    AND bd.author_id = a.author_id
    AND bd.publisher_id = p.publisher_id;

  SELECT count(*) INTO present_in_borrow FROM borrow_a_book
    WHERE book_id = bookId AND status = "ACTIVE" AND return_datetime IS NULL;

SELECT count(*) INTO present_in_reserve FROM reserve_a_book
    WHERE book_id = bookId AND status = "ACTIVE";
END
$$

delimiter ;
```

6.Print the book id and titles of books published by a publisher.



```sql
-------------------------------------------------------------------------------------
-- Print the book id and titles of books published by a publisher

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`print_bookIds_title_by_publisher_name` $$
CREATE PROCEDURE `DBProject`.`print_bookIds_title_by_publisher_name` (IN  bookPublisherName varchar(20))
BEGIN

SELECT bk.book_id, bkd.title
FROM book bk, book_details bkd
WHERE bkd.ISBN IN (SELECT bd.ISBN
  FROM book_details bd, publishers p
    WHERE p.publisher_name LIKE CONCAT('%',bookPublisherName,'%')
    AND bd.publisher_id = p.publisher_id)
    AND bk.ISBN = bkd.ISBN;
END
$$


delimiter ;
```

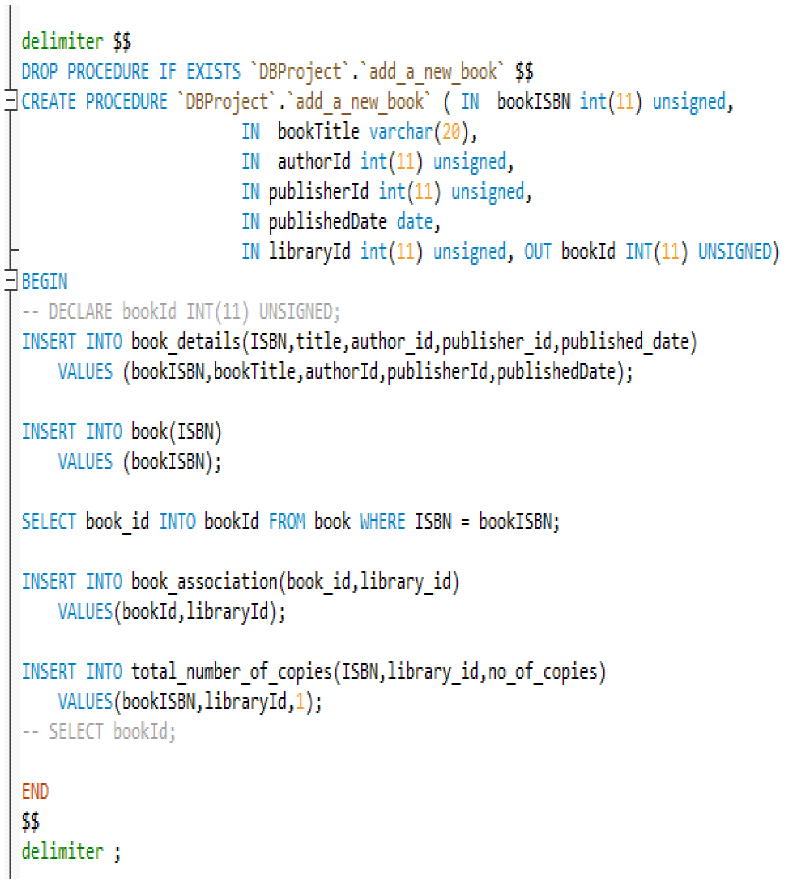
7. Check Admin and Reader (Validation Purpose) :



```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`check_reader` $$
CREATE PROCEDURE `DBProject`.`check_reader` (IN  readerId bigint unsigned,
                          OUT present_in_reader int)
BEGIN
  SELECT count(*) INTO present_in_reader FROM reader_details
    WHERE reader_id = readerId;
END
$$

delimiter ;

-- --------------------------------------------------------------------------

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`check_admin` $$
CREATE PROCEDURE `DBProject`.`check_admin` (IN  adminId int(11) unsigned,
                  IN  adminPassword varchar(20),
                              OUT present_in_admin int)
BEGIN
  SELECT count(*) INTO present_in_admin FROM admin
    WHERE admin_id = adminId AND admin_password = adminPassword;
END
$$

delimiter ;

-- ---------------------------------------------------------------------
```

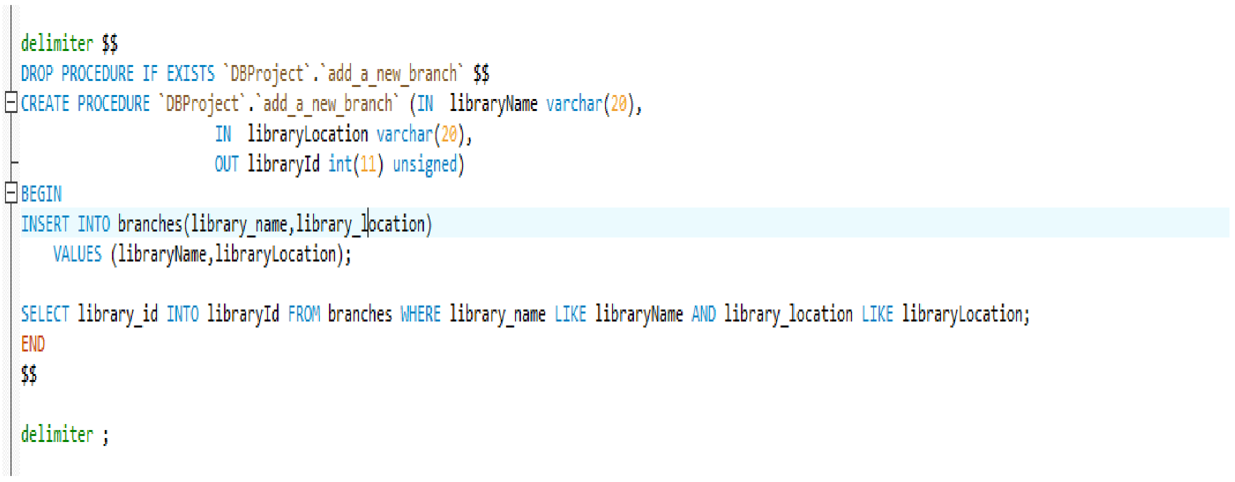
Admin Functionality

1. **Add a book and Add new branch**



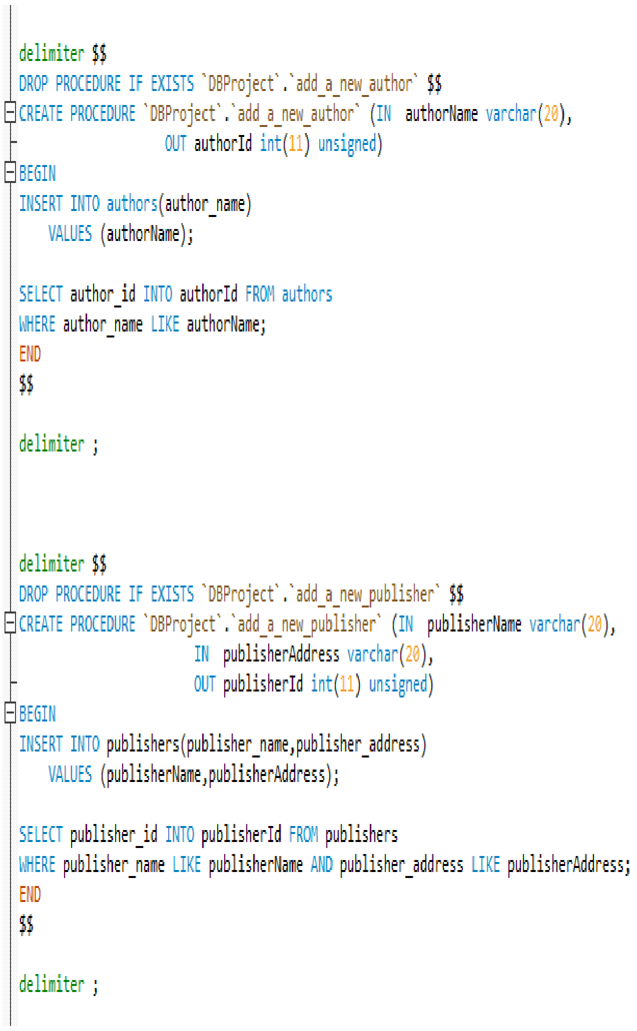
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_book` $$
CREATE PROCEDURE `DBProject`.`add_a_new_book` ( IN  bookISBN int(11) unsigned,
                        IN  bookTitle varchar(20),
                        IN  authorId int(11) unsigned,
                        IN publisherId int(11) unsigned,
                        IN publishedDate date,
                        IN libraryId int(11) unsigned, OUT bookId INT(11) UNSIGNED)
BEGIN
-- DECLARE bookId INT(11) UNSIGNED;
INSERT INTO book_details(ISBN,title,author_id,publisher_id,published_date)
    VALUES (bookISBN,bookTitle,authorId,publisherId,publishedDate);

INSERT INTO book(ISBN)
    VALUES (bookISBN);

SELECT book_id INTO bookId FROM book WHERE ISBN = bookISBN;

INSERT INTO book_association(book_id,library_id)
    VALUES(bookId,libraryId);

INSERT INTO total_number_of_copies(ISBN,library_id,no_of_copies)
    VALUES(bookISBN,libraryId,1);
-- SELECT bookId;

END
$$
delimiter ;
```

Add New Branch :



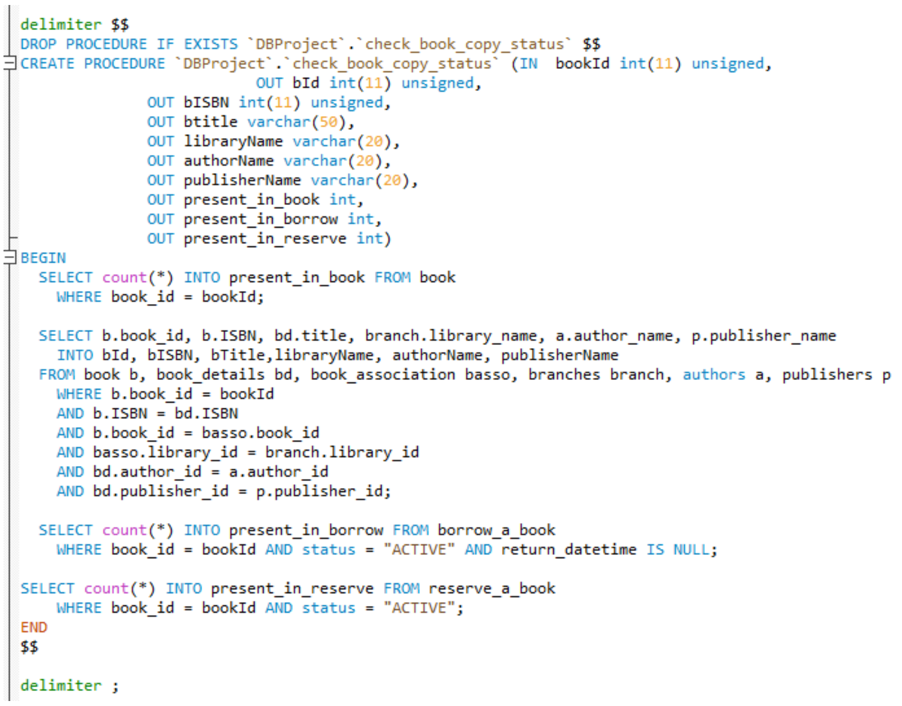
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_branch` $$
CREATE PROCEDURE `DBProject`.`add_a_new_branch` (IN  libraryName varchar(20),
                        IN  libraryLocation varchar(20),
                        OUT libraryId int(11) unsigned)
BEGIN
INSERT INTO branches(library_name,library_location)
    VALUES (libraryName,libraryLocation);

SELECT library_id INTO libraryId FROM branches WHERE library_name LIKE libraryName AND library_location LIKE libraryLocation;
END
$$

delimiter ;
```

Add New Author and Publisher



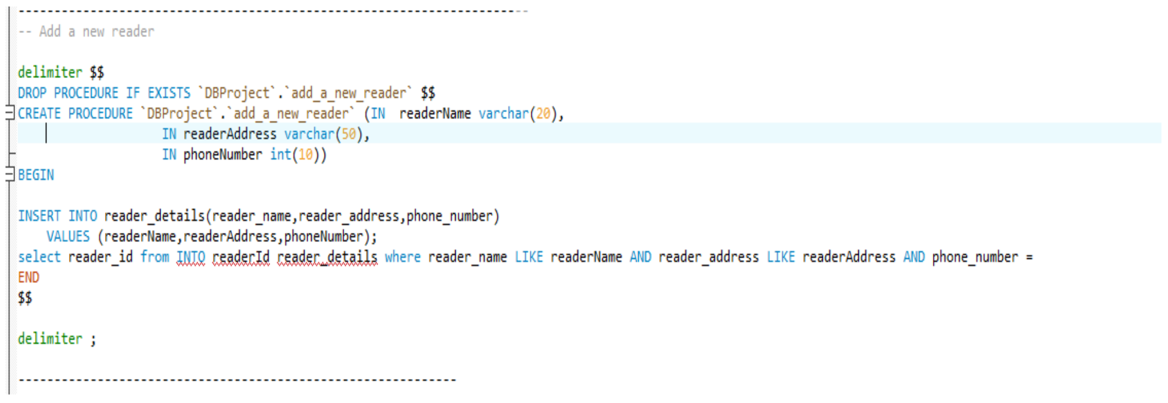
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_author` $$
CREATE PROCEDURE `DBProject`.`add_a_new_author` (IN  authorName varchar(20),
                    OUT authorId int(11) unsigned)
BEGIN
INSERT INTO authors(author_name)
    VALUES (authorName);

SELECT author_id INTO authorId FROM authors
WHERE author_name LIKE authorName;
END
$$

delimiter ;



delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_publisher` $$
CREATE PROCEDURE `DBProject`.`add_a_new_publisher` (IN  publisherName varchar(20),
                        IN  publisherAddress varchar(20),
                        OUT publisherId int(11) unsigned)
BEGIN
INSERT INTO publishers(publisher_name,publisher_address)
    VALUES (publisherName,publisherAddress);

SELECT publisher_id INTO publisherId FROM publishers
WHERE publisher_name LIKE publisherName AND publisher_address LIKE publisherAddress;
END
$$

delimiter ;
```

2. Search book copy and check its status



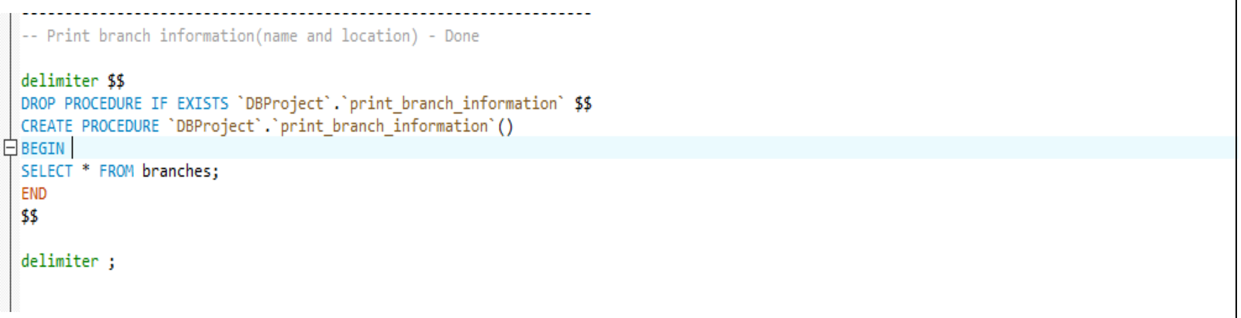
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`check_book_copy_status` $$
CREATE PROCEDURE `DBProject`.`check_book_copy_status` (IN  bookId int(11) unsigned,
                          OUT bId int(11) unsigned,
              OUT bISBN int(11) unsigned,
              OUT btitle varchar(50),
              OUT libraryName varchar(20),
              OUT authorName varchar(20),
              OUT publisherName varchar(20),
              OUT present_in_book int,
              OUT present_in_borrow int,
              OUT present_in_reserve int)
BEGIN
  SELECT count(*) INTO present_in_book FROM book
    WHERE book_id = bookId;

  SELECT b.book_id, b.ISBN, bd.title, branch.library_name, a.author_name, p.publisher_name
    INTO bId, bISBN, bTitle,libraryName, authorName, publisherName
  FROM book b, book_details bd, book_association basso, branches branch, authors a, publishers p
    WHERE b.book_id = bookId
    AND b.ISBN = bd.ISBN
    AND b.book_id = basso.book_id
    AND basso.library_id = branch.library_id
    AND bd.author_id = a.author_id
    AND bd.publisher_id = p.publisher_id;

  SELECT count(*) INTO present_in_borrow FROM borrow_a_book
    WHERE book_id = bookId AND status = "ACTIVE" AND return_datetime IS NULL;

SELECT count(*) INTO present_in_reserve FROM reserve_a_book
    WHERE book_id = bookId AND status = "ACTIVE";
END
$$

delimiter ;
```

3. Add new reader.

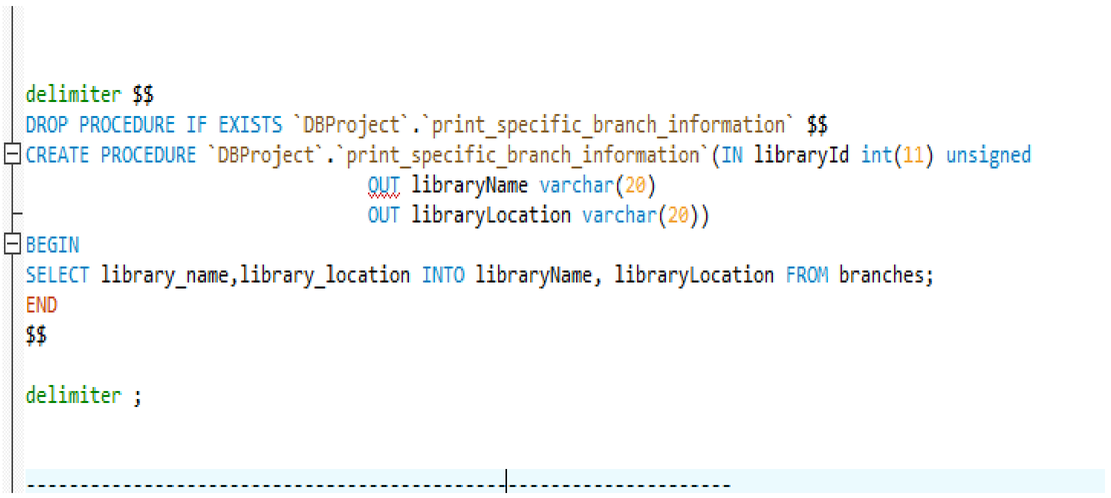


```
---------------------------------------------------------------------
-- Add a new reader

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_reader` $$
CREATE PROCEDURE `DBProject`.`add_a_new_reader` (IN  readerName varchar(20),
                  IN readerAddress varchar(50),
                  IN phoneNumber int(10))
BEGIN

INSERT INTO reader_details(reader_name,reader_address,phone_number)
    VALUES (readerName,readerAddress,phoneNumber);
select reader_id from INTO readerId reader_details where reader_name LIKE readerName AND reader_address LIKE readerAddress AND phone_number =
END
$$

delimiter ;

-----------------------------------------------------------
```

4. Print branch information (name and location) and with Specific Branch Information :

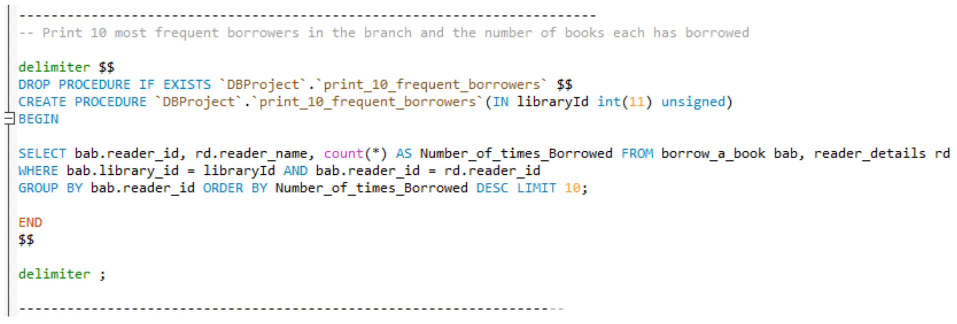


```
-------------------------------------------------------------------
-- Print branch information(name and location) - Done

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`print_branch_information` $$
CREATE PROCEDURE `DBProject`.`print_branch_information`()
BEGIN
SELECT * FROM branches;
END
$$

delimiter ;
```



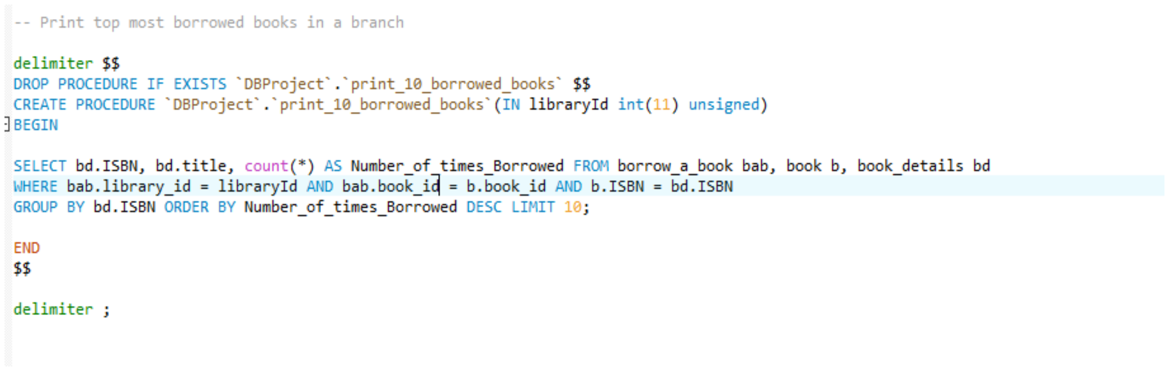
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`print_specific_branch_information` $$
CREATE PROCEDURE `DBProject`.`print_specific_branch_information`(IN libraryId int(11) unsigned
                                OUT libraryName varchar(20)
                                OUT libraryLocation varchar(20))
BEGIN
SELECT library_name,library_location INTO libraryName, libraryLocation FROM branches;
END
$$

delimiter ;

------------------------------------------------------------------
```

5. **Print top 10 most frequent borrowers in a branch and the number of books each has borrowed.**



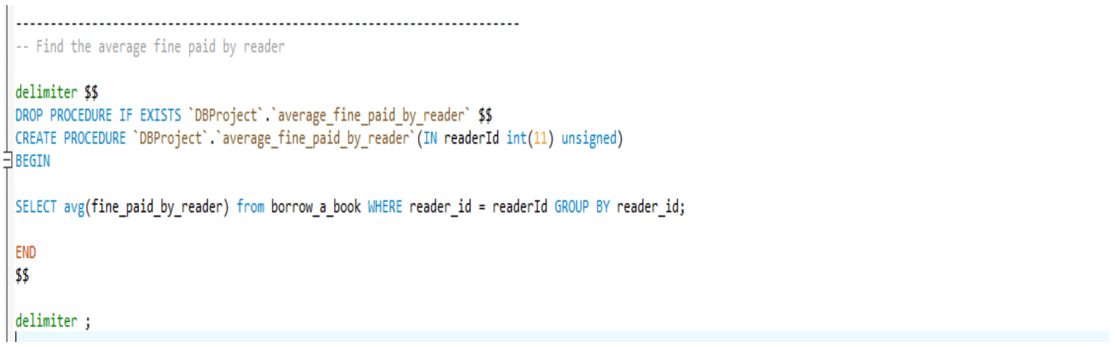
```
-------------------------------------------------------------------------
-- Print 10 most frequent borrowers in the branch and the number of books each has borrowed

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`print_10_frequent_borrowers` $$
CREATE PROCEDURE `DBProject`.`print_10_frequent_borrowers`(IN libraryId int(11) unsigned)
BEGIN

SELECT bab.reader_id, rd.reader_name, count(*) AS Number_of_times_Borrowed FROM borrow_a_book bab, reader_details rd
WHERE bab.library_id = libraryId AND bab.reader_id = rd.reader_id
GROUP BY bab.reader_id ORDER BY Number_of_times_Borrowed DESC LIMIT 10;

END
$$

delimiter ;

---------------------------------------------------------------------
```

6. **Print top 10 most borrowed books in a branch.**



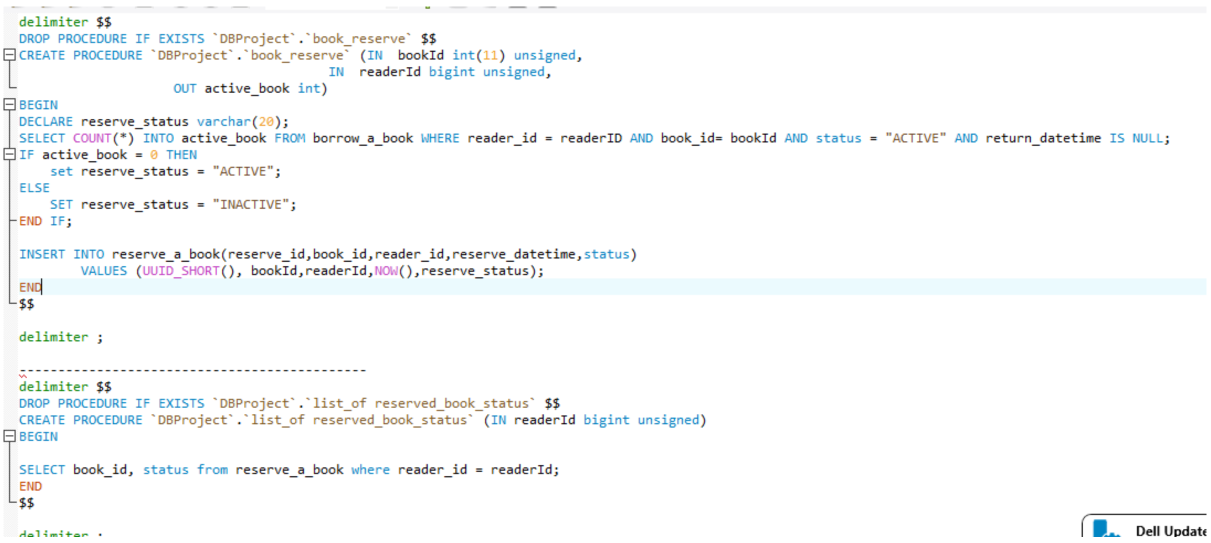
```
-- Print top most borrowed books in a branch

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`print_10_borrowed_books` $$
CREATE PROCEDURE `DBProject`.`print_10_borrowed_books`(IN libraryId int(11) unsigned)
BEGIN

SELECT bd.ISBN, bd.title, count(*) AS Number_of_times_Borrowed FROM borrow_a_book bab, book b, book_details bd
WHERE bab.library_id = libraryId AND bab.book_id = b.book_id AND b.ISBN = bd.ISBN
GROUP BY bd.ISBN ORDER BY Number_of_times_Borrowed DESC LIMIT 10;

END
$$

delimiter ;
```

7. Find the average fine paid per reader.



```
------------------------------------------------------------------------
-- Find the average fine paid by reader

delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`average_fine_paid_by_reader` $$
CREATE PROCEDURE `DBProject`.`average_fine_paid_by_reader`(IN readerId int(11) unsigned)
BEGIN

SELECT avg(fine_paid_by_reader) from borrow_a_book WHERE reader_id = readerId GROUP BY reader_id;

END
$$

delimiter ;
```

8. Reserve a book.



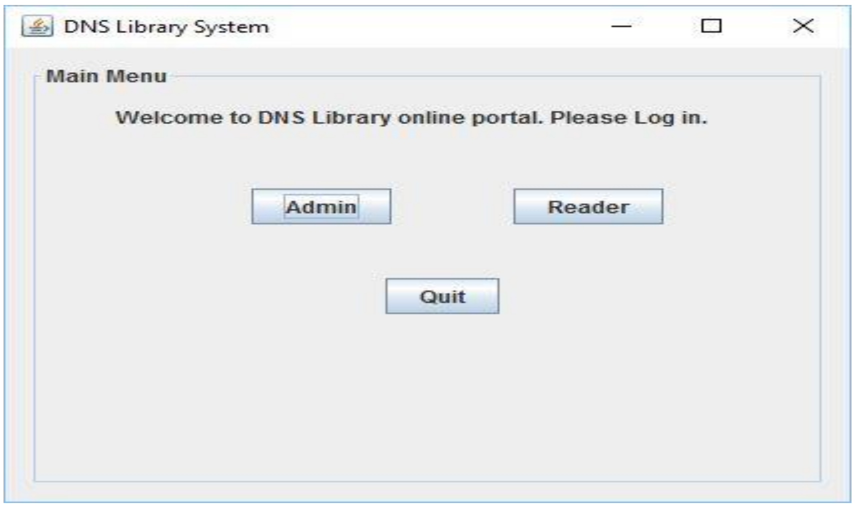
```
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`book_reserve` $$
CREATE PROCEDURE `DBProject`.`book_reserve` (IN  bookId int(11) unsigned,
                                        IN  readerId bigint unsigned,
                    OUT active_book int)
BEGIN
DECLARE reserve_status varchar(20);
SELECT COUNT(*) INTO active_book FROM borrow_a_book WHERE reader_id = readerID AND book_id= bookId AND status = "ACTIVE" AND return_datetime IS NULL;
IF active_book = 0 THEN
    set reserve_status = "ACTIVE";
ELSE
    SET reserve_status = "INACTIVE";
END IF;

INSERT INTO reserve_a_book(reserve_id,book_id,reader_id,reserve_datetime,status)
        VALUES (UUID_SHORT(), bookId,readerId,NOW(),reserve_status);
END
$$

delimiter ;

---------------------------------------------
delimiter $$
DROP PROCEDURE IF EXISTS `DBProject`.`list_of_reserved_book_status` $$
CREATE PROCEDURE `DBProject`.`list_of_reserved_book_status` (IN readerId bigint unsigned)
BEGIN

SELECT book_id, status from reserve_a_book where reader_id = readerId;
END
$$

delimiter ;
```

User Guide

Window snapshots of the use of the program for each function

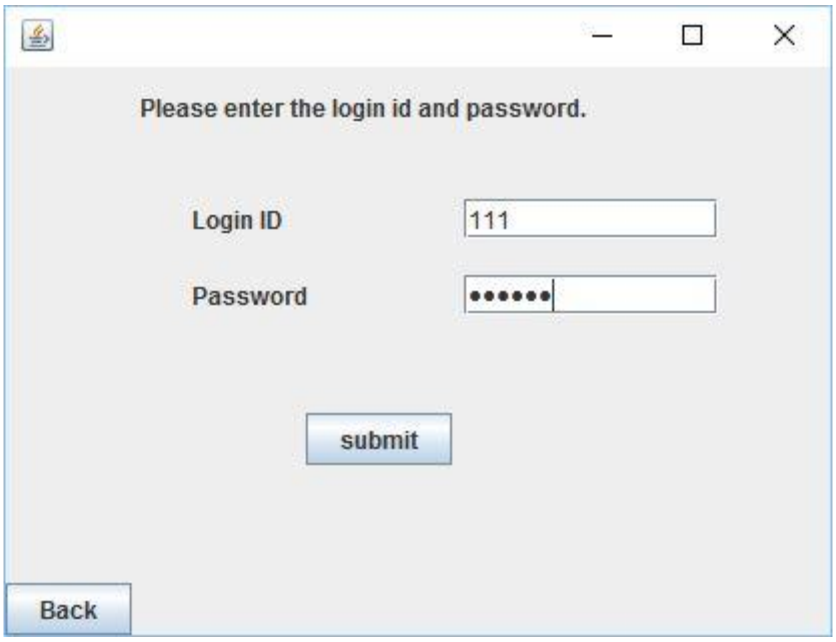
Admin

1. **Main Menu**

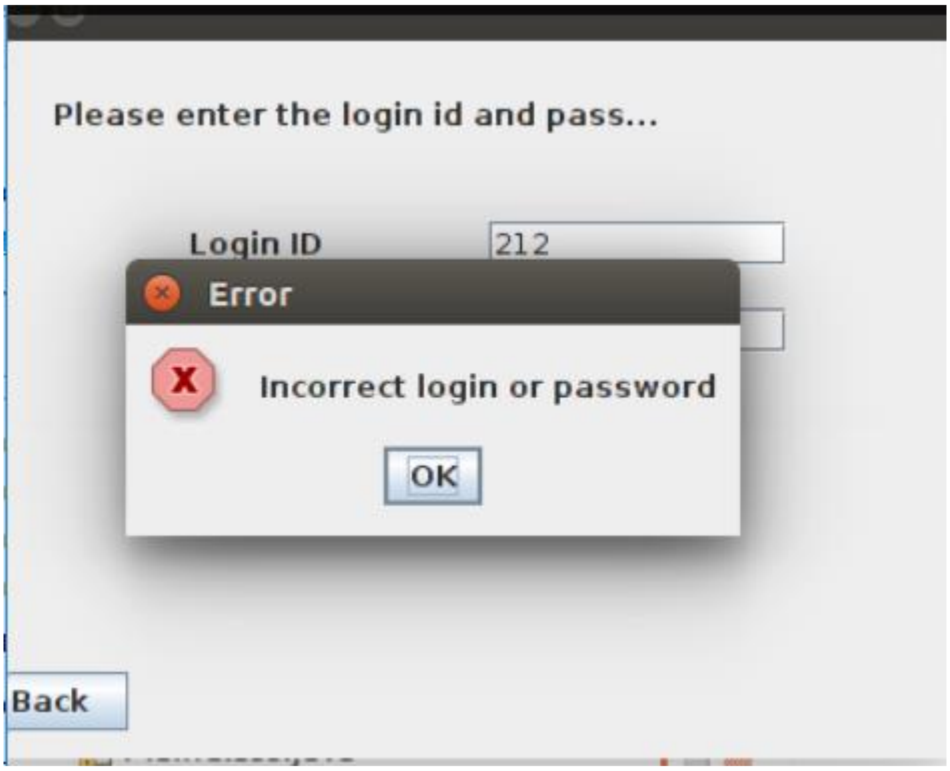


Now click on the Admin Button:

2. **Admin Login**



If we enter the wrong username or password or both


Please enter the login id and pass...

Login ID 212

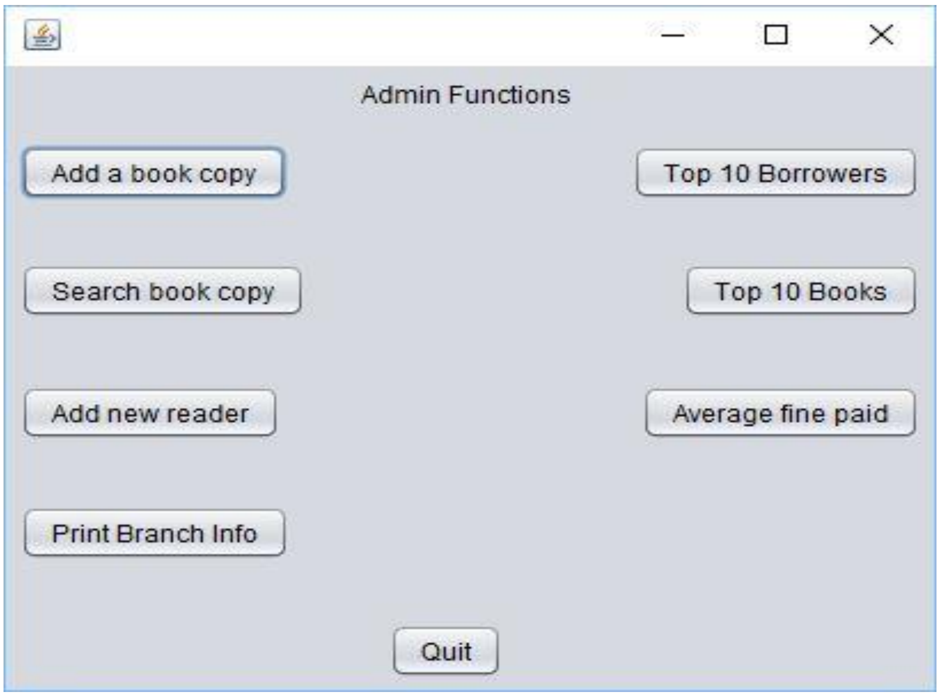
Error

Incorrect login or password

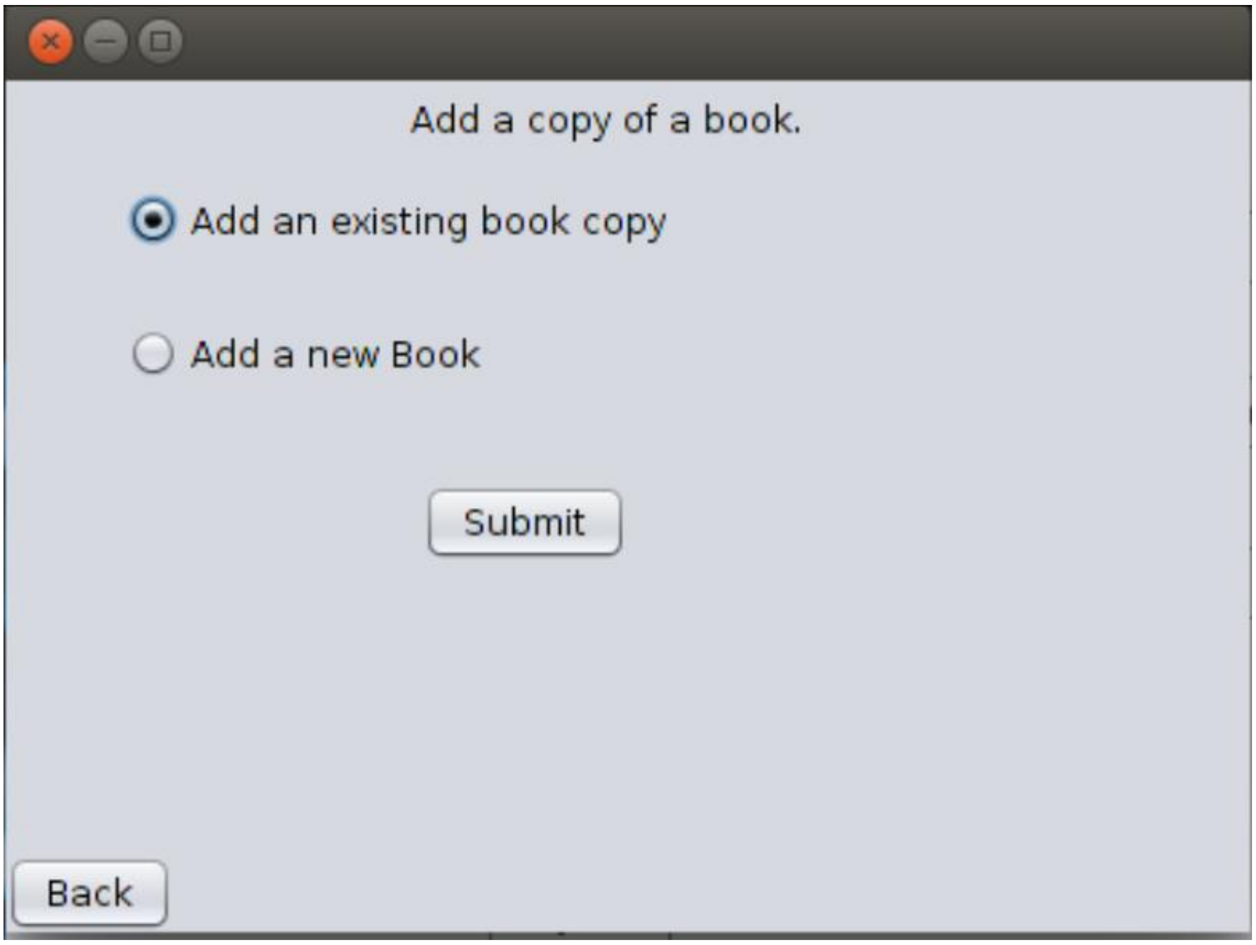
OK

Back

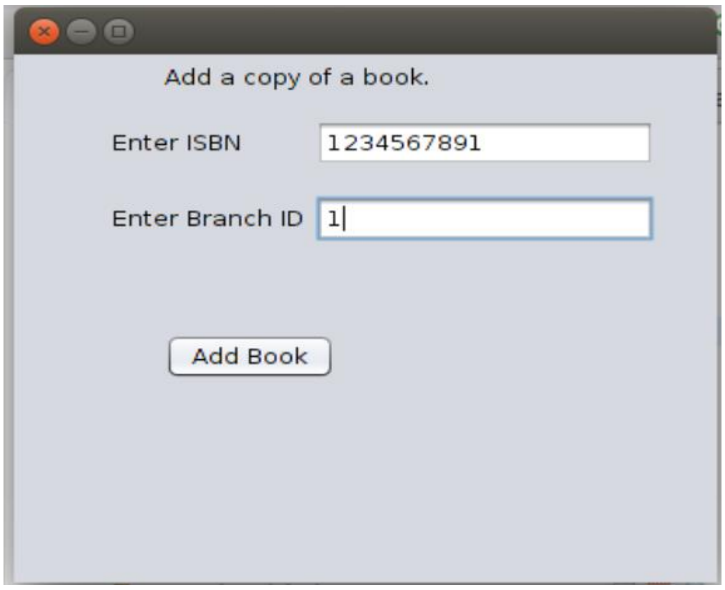
3. **Admin Features**


Admin Functions

Add a copy of a book



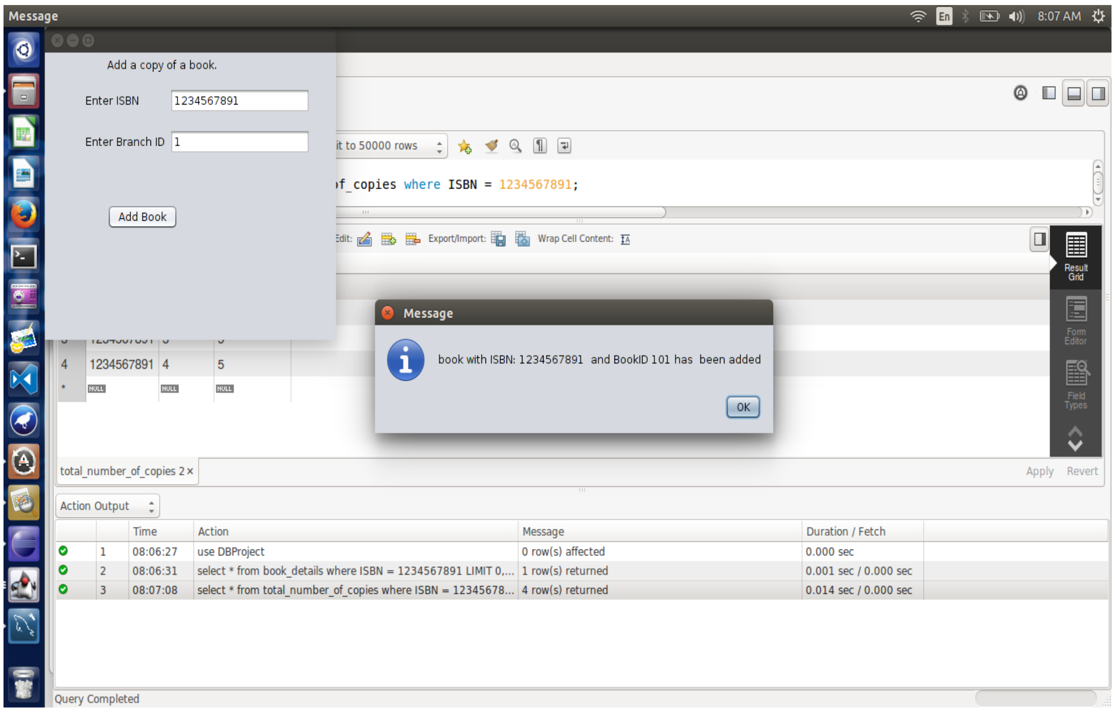
Now Selected Radio button for Add an existing copy of Book



Enter ISBN 1234567891

Enter Branch ID 1

Now you will see that the copy of book is added to the system means in Database:



Add a copy of a book.

Enter ISBN 1234567891

Enter Branch ID 1

```
of_copies where ISBN = 1234567891;
```

4	1234567891	4	5
*	NULL	NULL	NULL

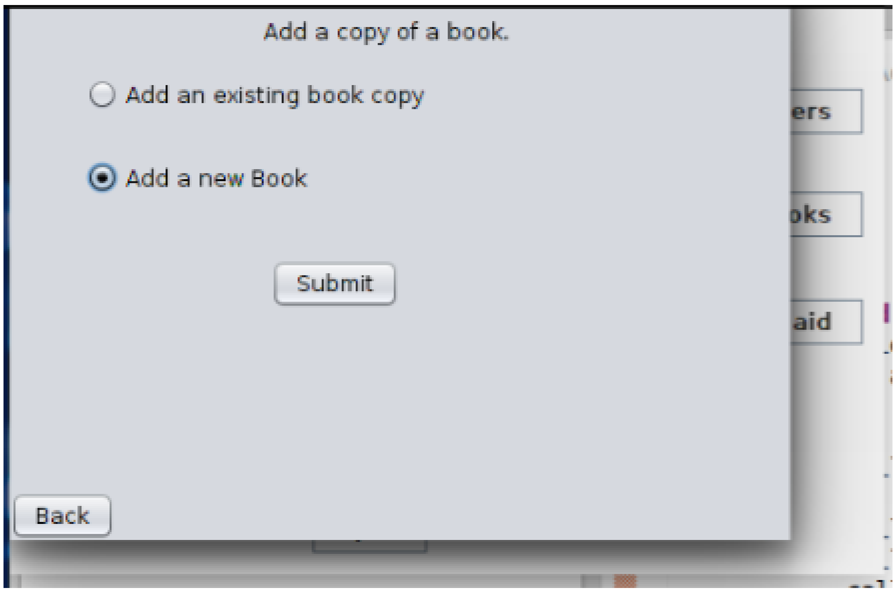
Message

book with ISBN: 1234567891 and BookID 101 has been added

```
     Time      Action                                                   Message            Duration / Fetch
  1  08:06:27  use DBProject                                            0 row(s) affected  0.000 sec
  2  08:06:31  select * from book_details where ISBN = 1234567891 LIMIT 0,...  1 row(s) returned  0.001 sec / 0.000 sec
  3  08:07:08  select * from total_number_of_copies where ISBN = 12345678...  4 row(s) returned  0.014 sec / 0.000 sec
```

Query Completed

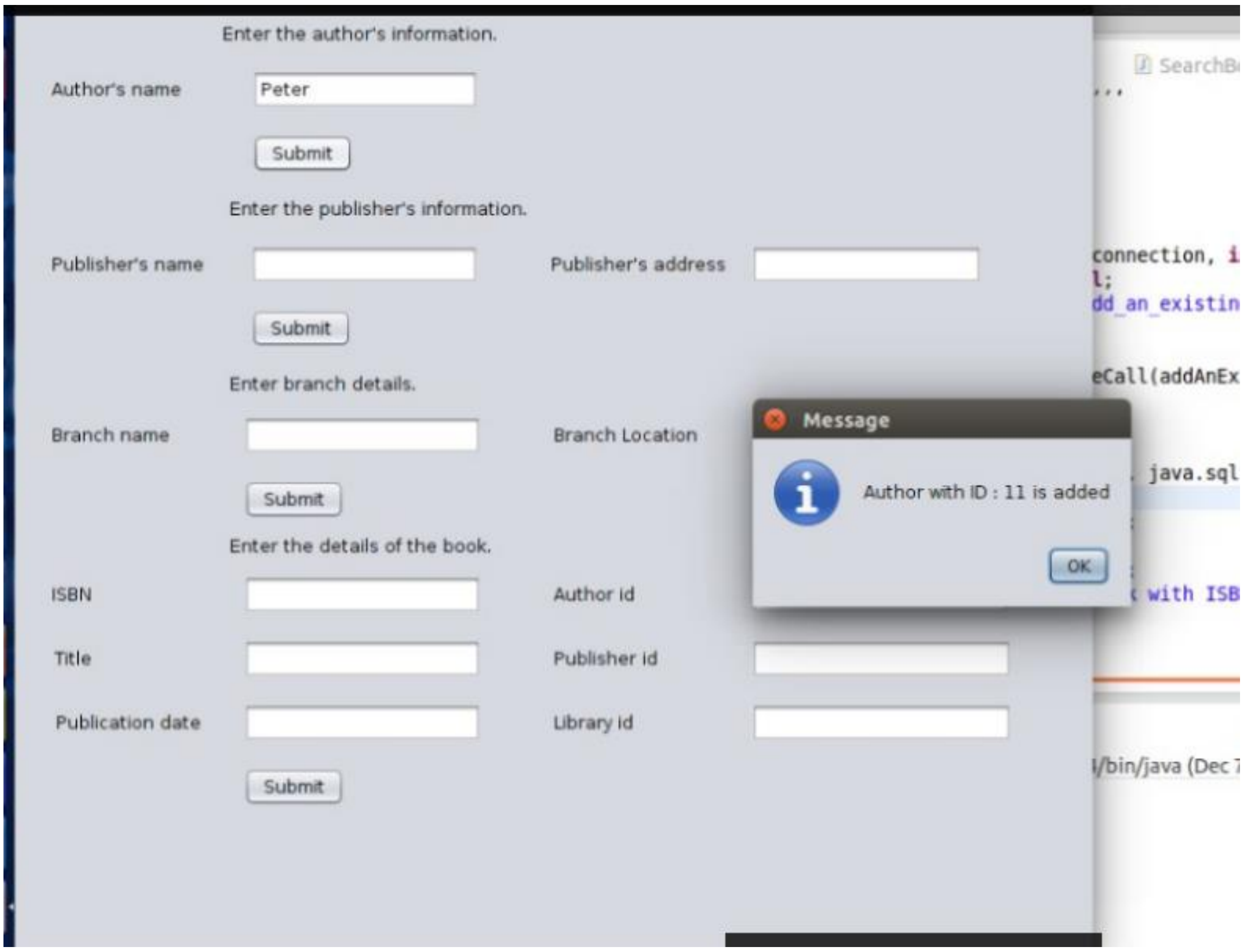
6. Select second option to add a new book :



Add a copy of a book.

Add an existing book copy [✓ Add a new Book]

Now you are about to see the screen as below option :


Enter the author's information.

Author's name: Peter

Submit

Enter the publisher's information.

Publisher's name

Publisher's address

Submit

Enter branch details.

Branch name

Branch Location

Submit

Enter the details of the book.

ISBN

Author id

Title

Publisher id

Publication date

Library id

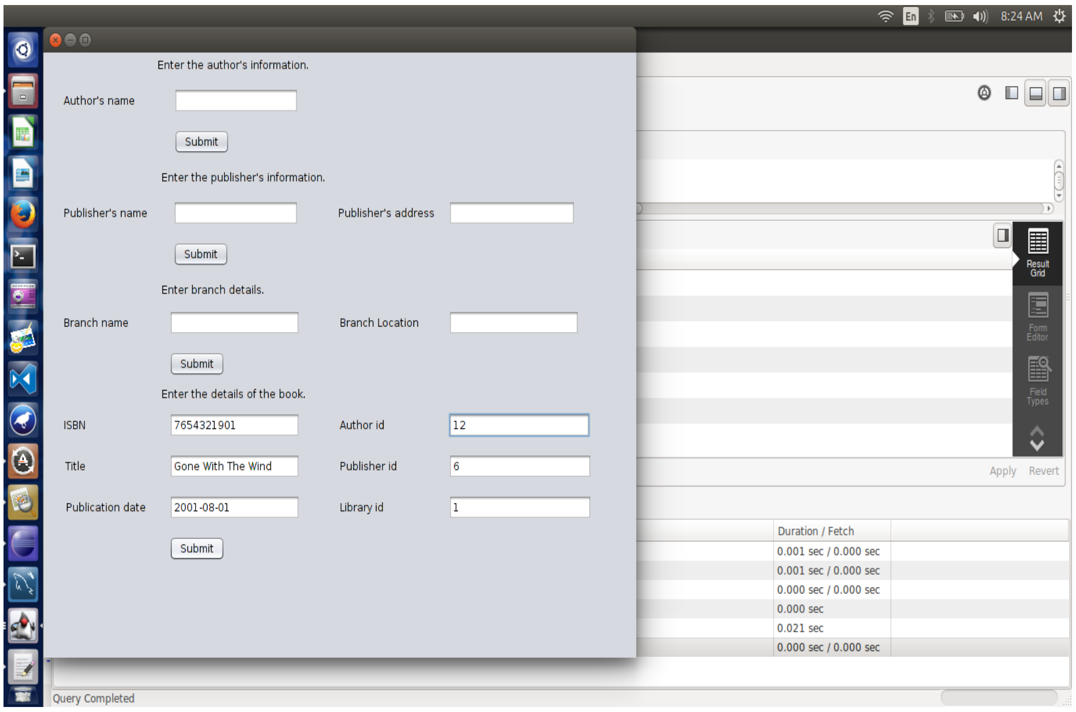
Submit

Message

Author with ID : 11 is added

OK

We have created one page where we are about to add different items from one page and adding the book Information which saved in the database.


Enter the author's information.

Author's name			
Publisher's name		Publisher's address	
Branch name		Branch Location	
ISBN	7654321901	Author id	12
Title	Gone With The Wind	Publisher id	6
Publication date	2001-08-01	Library id	1

Enter the publisher's information.

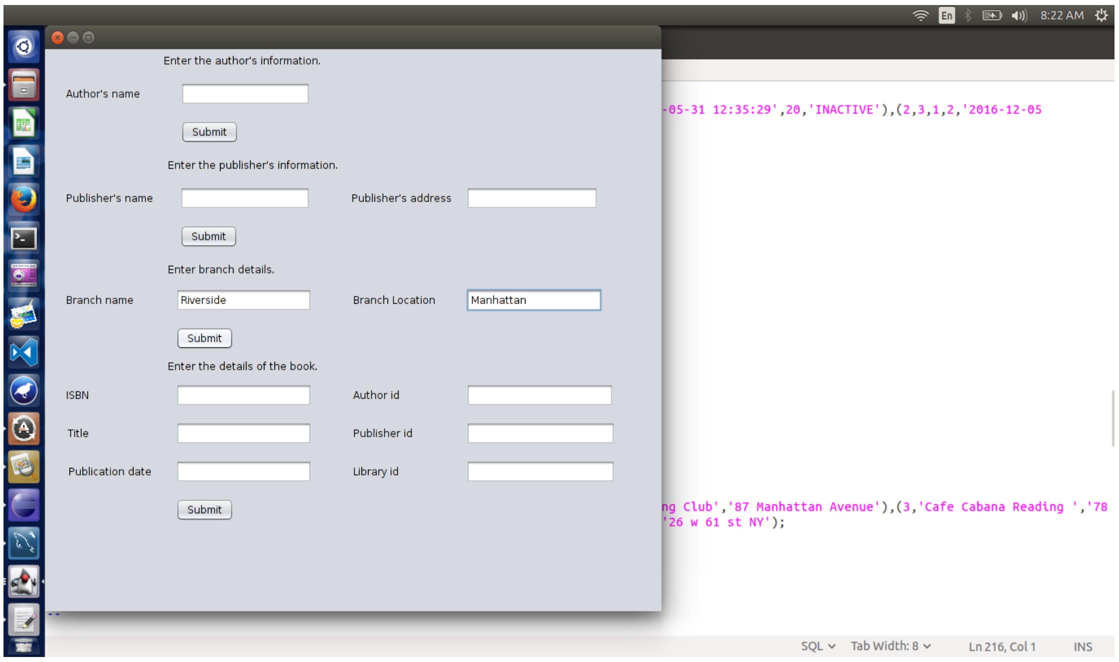
Enter branch details.

Enter the details of the book.

```
Duration / Fetch
0.001 sec / 0.000 sec
0.001 sec / 0.000 sec
0.000 sec / 0.000 sec
0.000 sec
0.021 sec
0.000 sec / 0.000 sec
```

Query Completed

The screen below shows that the branch for the new copy of a book should be entered in the given fields.


Enter the author's information.

Author's name			
Publisher's name		Publisher's address	
Branch name	Riverside	Branch Location	Manhattan
ISBN		Author id	
Title		Publisher id	
Publication date		Library id	

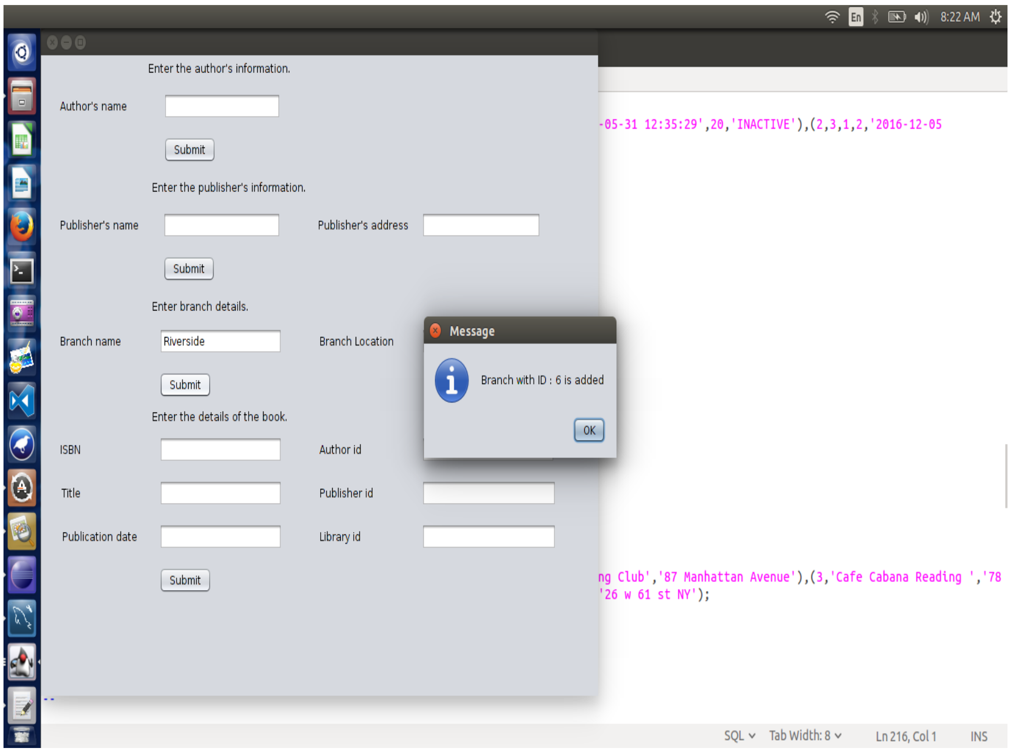
Enter the publisher's information.

Enter branch details.

Enter the details of the book.

```
-05-31 12:35:29',20,'INACTIVE'),(2,3,1,2,'2016-12-05

ng Club','87 Manhattan Avenue'),(3,'Cafe Cabana Reading ','78
'26 w 61 st NY');
```

After successfully adding the branch to the copy of the book, it shows a screen like this:



Enter the author's information.

Author's name

Submit

Enter the publisher's information.

Publisher's name | Publisher's address

Submit

Enter branch details.

Branch name: Riverside | Branch Location

Submit

Enter the details of the book.

ISBN | Author id

Title | Publisher id

Publication date | Library id

Submit

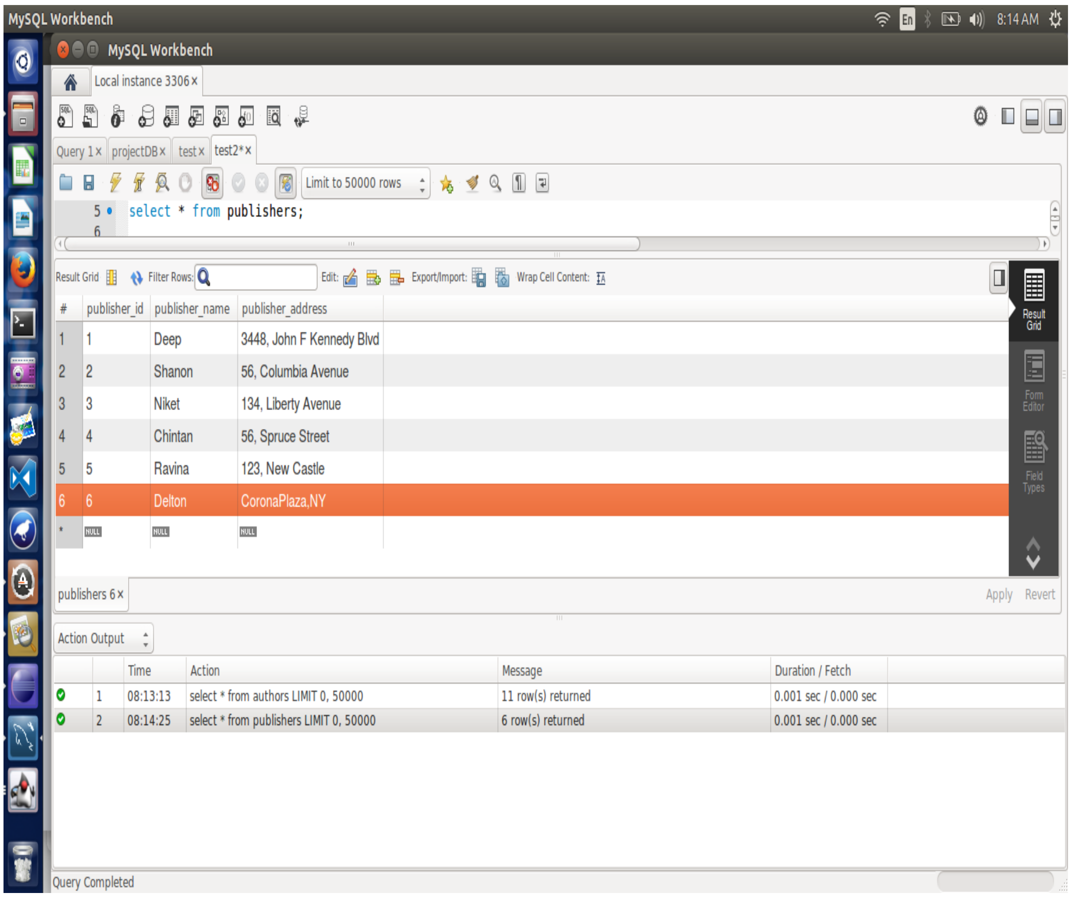
Message

Branch with ID : 6 is added

OK

```
-05-31 12:35:29',20,'INACTIVE'),(2,3,1,2,'2016-12-05
```

```
ng Club','87 Manhattan Avenue'),(3,'Cafe Cabana Reading ','78
'26 w 61 st NY');
```

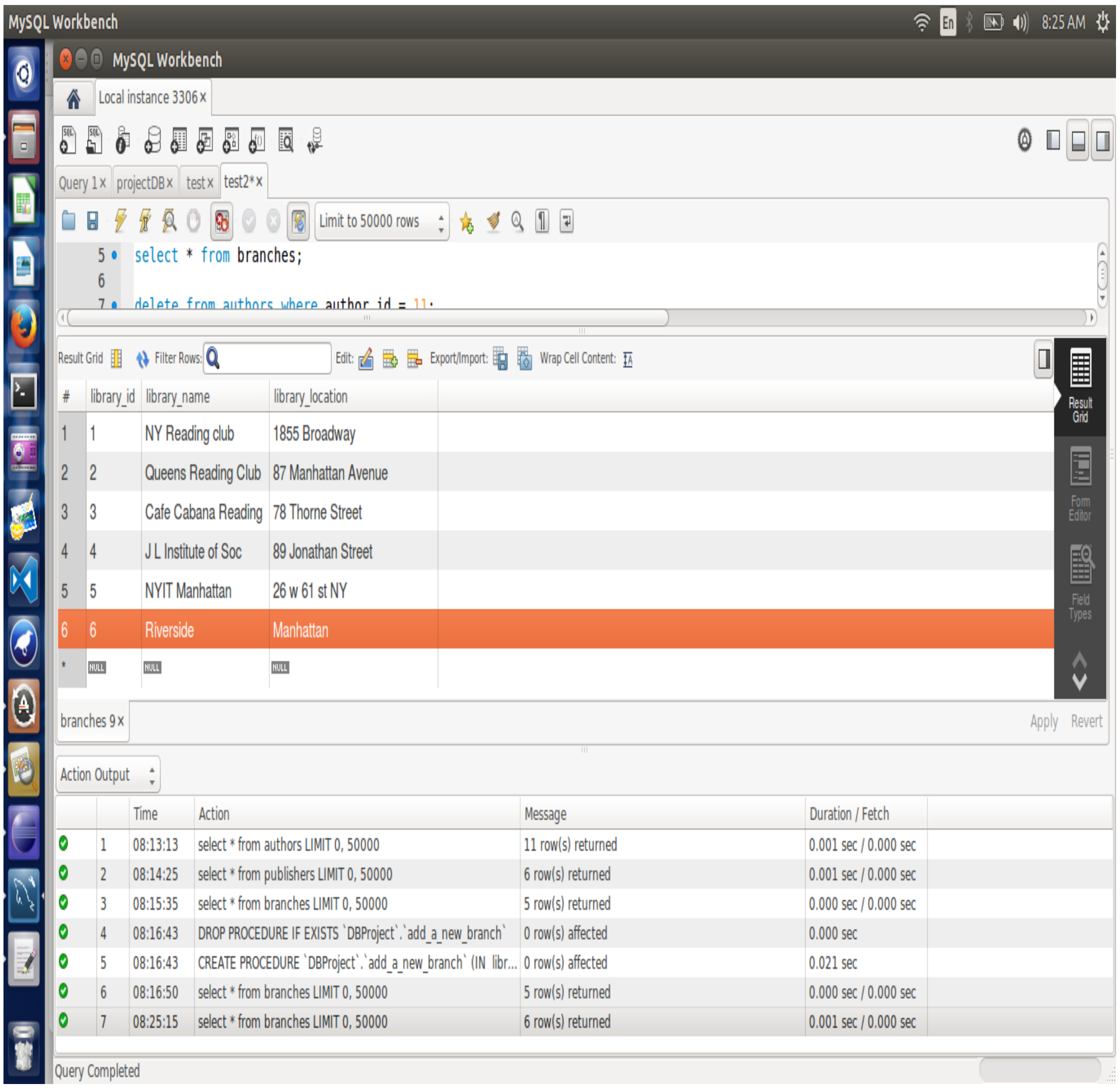


```
select * from publishers;
```

#	publisher_id	publisher_name	publisher_address
1	1	Deep	3448, John F Kennedy Blvd
2	2	Shanon	56, Columbia Avenue
3	3	Niket	134, Liberty Avenue
4	4	Chintan	56, Spruce Street
5	5	Ravina	123, New Castle
6	6	Delton	CoronaPlaza,NY
*	NULL	NULL	NULL

	Time	Action	Message	Duration / Fetch
1	08:13:13	select * from authors LIMIT 0, 50000	11 row(s) returned	0.001 sec / 0.000 sec
2	08:14:25	select * from publishers LIMIT 0, 50000	6 row(s) returned	0.001 sec / 0.000 sec

The screen above shows the table with the information of the readers.



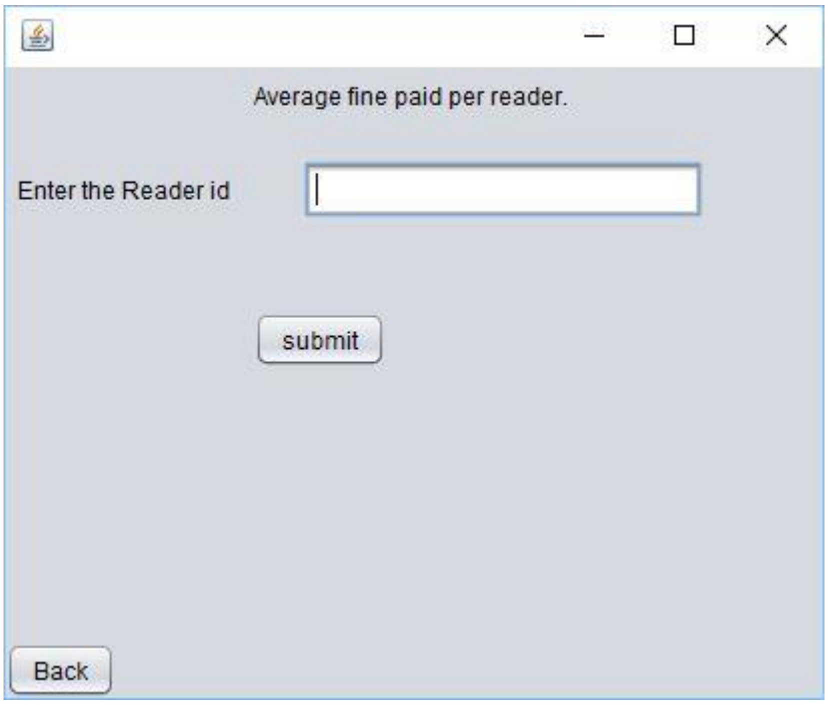
```
select * from branches;

delete from authors where author_id = 11;
```

#	library_id	library_name	library_location
1	1	NY Reading club	1855 Broadway
2	2	Queens Reading Club	87 Manhattan Avenue
3	3	Cafe Cabana Reading	78 Thorne Street
4	4	J L Institute of Soc	89 Jonathan Street
5	5	NYIT Manhattan	26 w 61 st NY
6	6	Riverside	Manhattan
*	NULL	NULL	NULL

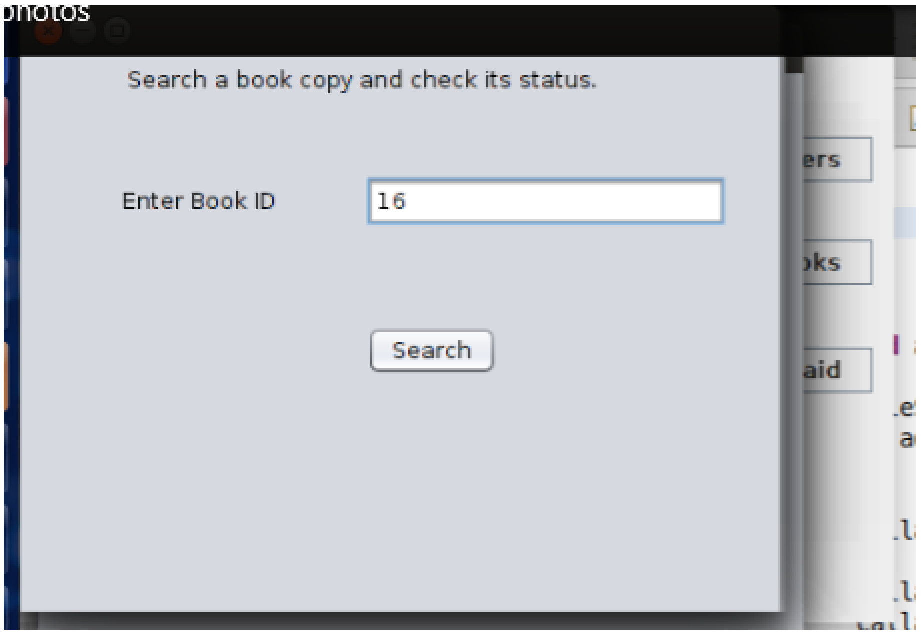
		Time	Action	Message	Duration / Fetch
✓	1	08:13:13	select * from authors LIMIT 0, 50000	11 row(s) returned	0.001 sec / 0.000 sec
✓	2	08:14:25	select * from publishers LIMIT 0, 50000	6 row(s) returned	0.001 sec / 0.000 sec
✓	3	08:15:35	select * from branches LIMIT 0, 50000	5 row(s) returned	0.000 sec / 0.000 sec
✓	4	08:16:43	DROP PROCEDURE IF EXISTS `DBProject`.`add_a_new_branch`	0 row(s) affected	0.000 sec
✓	5	08:16:43	CREATE PROCEDURE `DBProject`.`add_a_new_branch` (IN libr...	0 row(s) affected	0.021 sec
✓	6	08:16:50	select * from branches LIMIT 0, 50000	5 row(s) returned	0.000 sec / 0.000 sec
✓	7	08:25:15	select * from branches LIMIT 0, 50000	6 row(s) returned	0.001 sec / 0.000 sec

The screen above shows the information about the branches of the library.

7. Average fine paid per reader

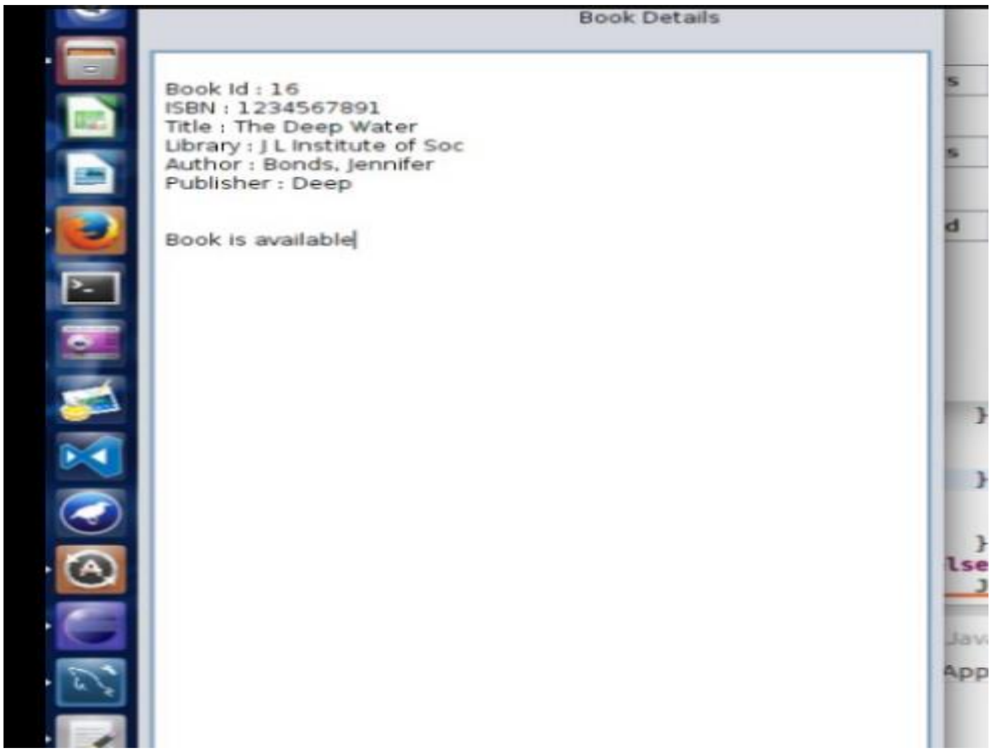


8. Search a book copy and check it's status



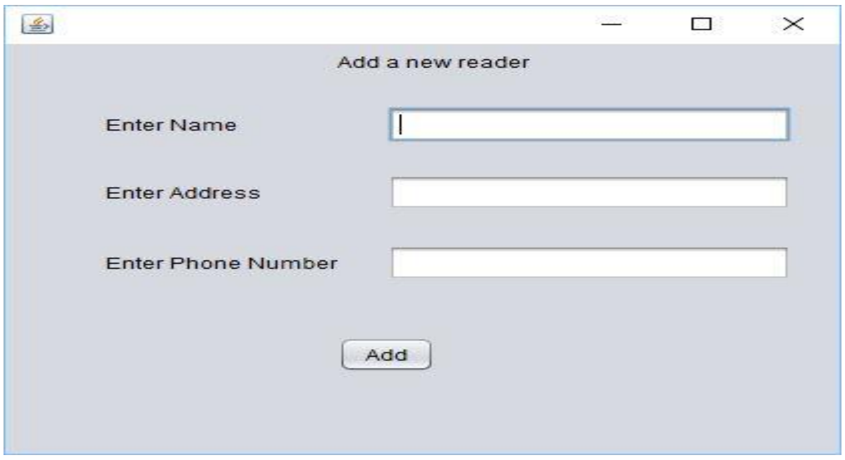
Enter Book ID: 16

Its printing below information:

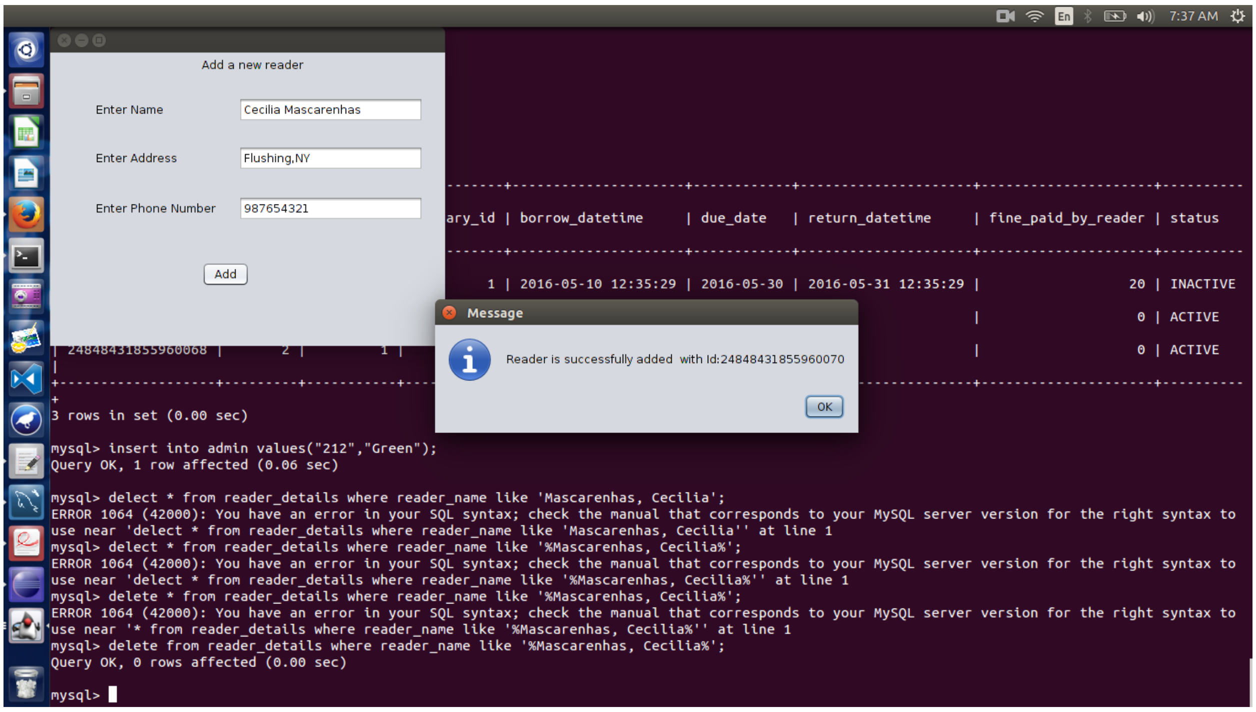


```
Book Id : 16
ISBN : 1234567891
Title : The Deep Water
Library : J L Institute of Soc
Author : Bonds, Jennifer
Publisher : Deep

Book is available
```

8. Add a new reader



The screen below shows that a reader has successfully been added to the database.


Add a new reader

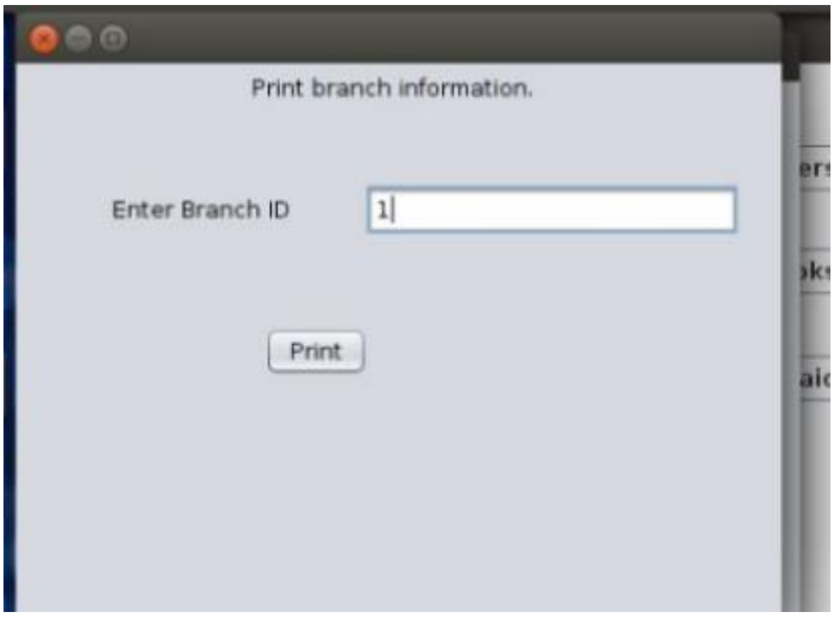
Enter Name	Cecilia Mascarenhas
Enter Address	Flushing,NY
Enter Phone Number	987654321

Message

Reader is successfully added with Id:24848431855960070

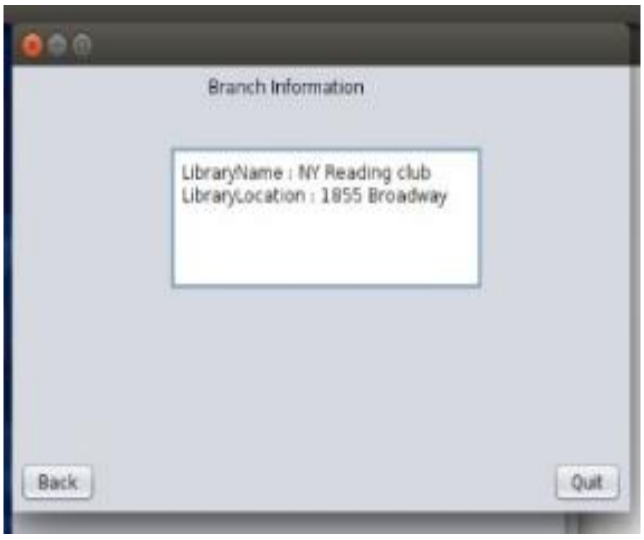
```
-------+--------------------+-----------+--------------------+--------------------+---------
ary_id | borrow_datetime    | due_date  | return_datetime    | fine_paid_by_reader | status
-------+--------------------+-----------+--------------------+--------------------+---------
     1 | 2016-05-10 12:35:29 | 2016-05-30 | 2016-05-31 12:35:29 |                 20 | INACTIVE
                                                              |                  0 | ACTIVE
| 24848431855960068 |        2 |          1 |                   |                  0 | ACTIVE
|
+------------------+--------+----------+---                  -----------+--------------------+---------
+
3 rows in set (0.00 sec)

mysql> insert into admin values("212","Green");
Query OK, 1 row affected (0.06 sec)

mysql> delect * from reader_details where reader_name like 'Mascarenhas, Cecilia';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to
use near 'delect * from reader_details where reader_name like 'Mascarenhas, Cecilia'' at line 1
mysql> delect * from reader_details where reader_name like '%Mascarenhas, Cecilia%';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to
use near 'delect * from reader_details where reader_name like '%Mascarenhas, Cecilia%'' at line 1
mysql> delete * from reader_details where reader_name like '%Mascarenhas, Cecilia%';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to
use near '* from reader_details where reader_name like '%Mascarenhas, Cecilia%'' at line 1
mysql> delete from reader_details where reader_name like '%Mascarenhas, Cecilia%';
Query OK, 0 rows affected (0.00 sec)

mysql> 
```

10. Print Branch Information


Print branch information.

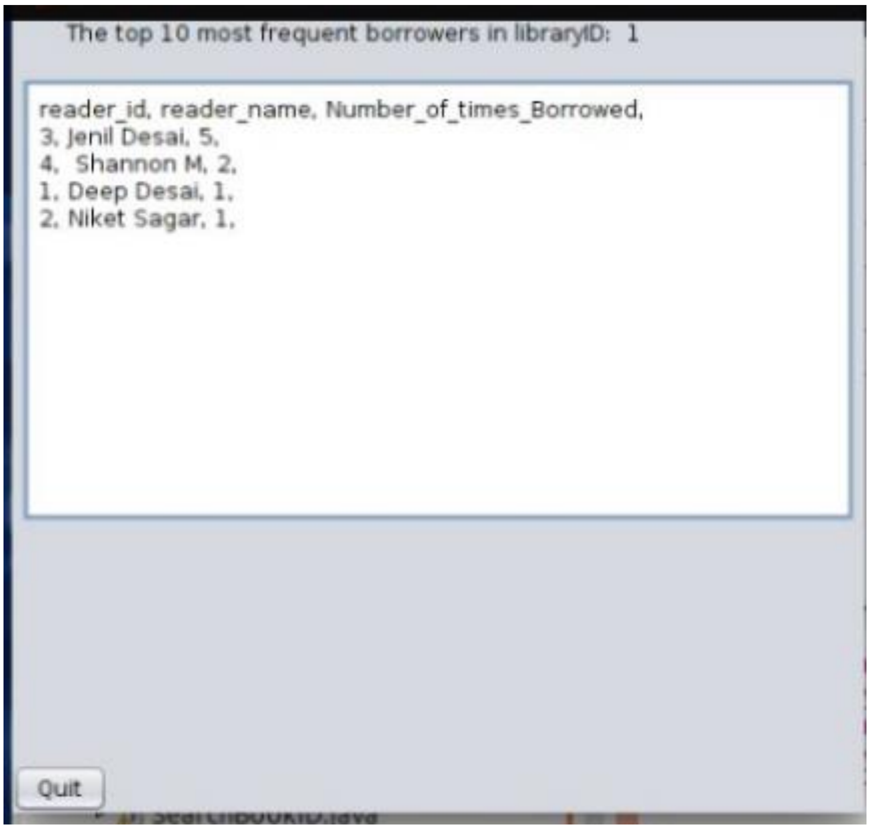
Enter Branch ID	1

Now you can see the branch information below :



```
LibraryName : NY Reading club
LibraryLocation : 1855 Broadway
```

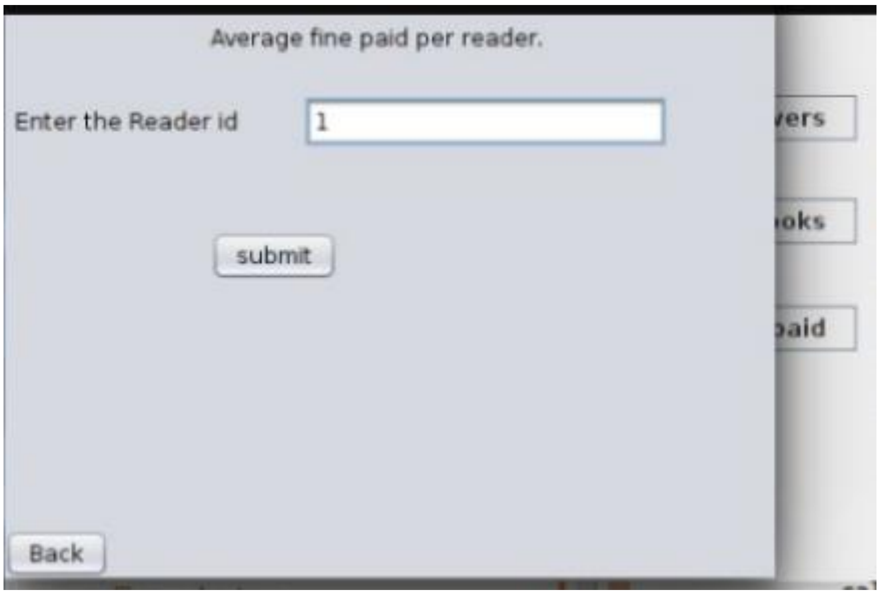
11. The top 10 most frequent borrowers in library ID : 1

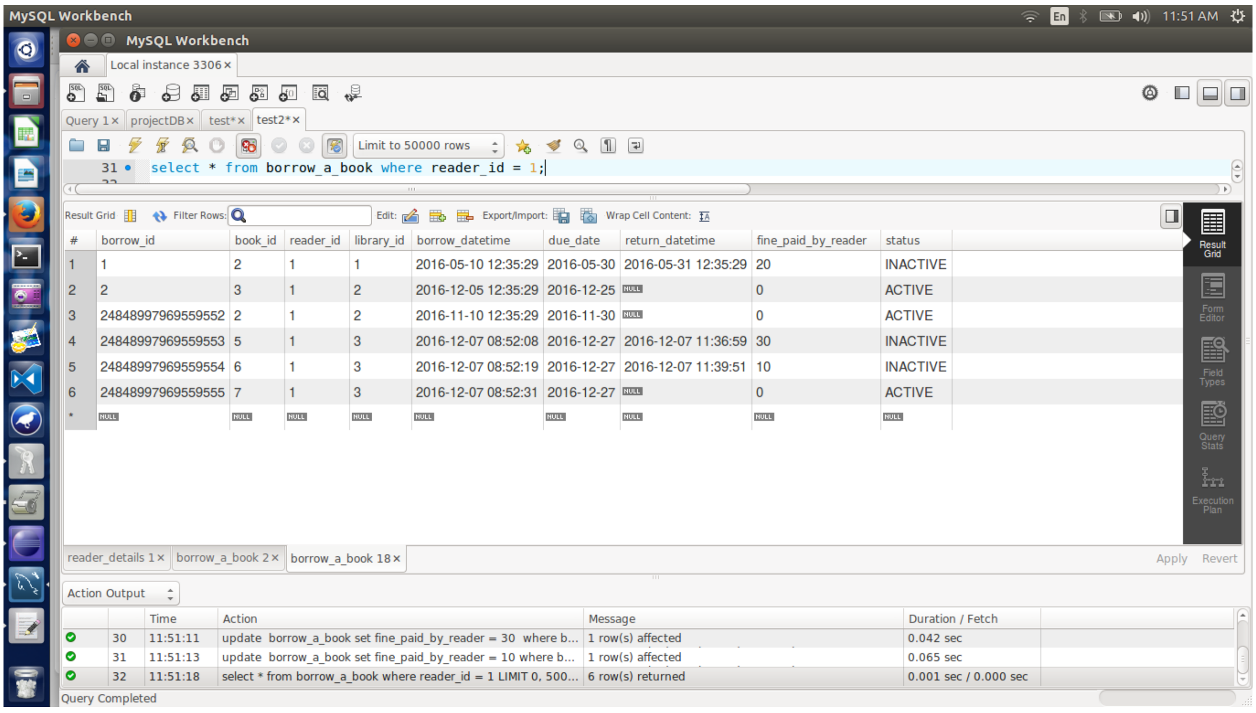


```
reader_id, reader_name, Number_of_times_Borrowed,
3, Jenil Desai, 5,
4,  Shannon M, 2,
1, Deep Desai, 1,
2, Niket Sagar, 1,
```

This how they add the reader information.

15. Average Fine Paid per reader



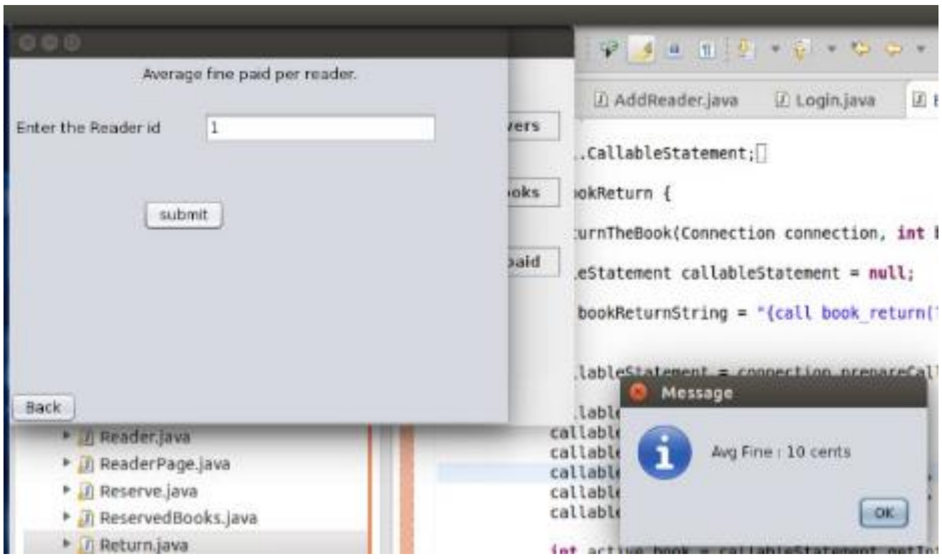


```
select * from borrow_a_book where reader_id = 1;
```

#	borrow_id	book_id	reader_id	library_id	borrow_datetime	due_date	return_datetime	fine_paid_by_reader	status
1	1	2	1	1	2016-05-10 12:35:29	2016-05-30	2016-05-31 12:35:29	20	INACTIVE
2	2	3	1	2	2016-12-05 12:35:29	2016-12-25	NULL	0	ACTIVE
3	24848997969559552	2	1	2	2016-11-10 12:35:29	2016-11-30	NULL	0	ACTIVE
4	24848997969559553	5	1	3	2016-12-07 08:52:08	2016-12-27	2016-12-07 11:36:59	30	INACTIVE
5	24848997969559554	6	1	3	2016-12-07 08:52:19	2016-12-27	2016-12-07 11:39:51	10	INACTIVE
6	24848997969559555	7	1	3	2016-12-07 08:52:31	2016-12-27	NULL	0	ACTIVE
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

		Time	Action	Message	Duration / Fetch
✓	30	11:51:11	update borrow_a_book set fine_paid_by_reader = 30 where b...	1 row(s) affected	0.042 sec
✓	31	11:51:13	update borrow_a_book set fine_paid_by_reader = 10 where b...	1 row(s) affected	0.065 sec
✓	32	11:51:18	select * from borrow_a_book where reader_id = 1 LIMIT 0, 500...	6 row(s) returned	0.001 sec / 0.000 sec

Query Completed

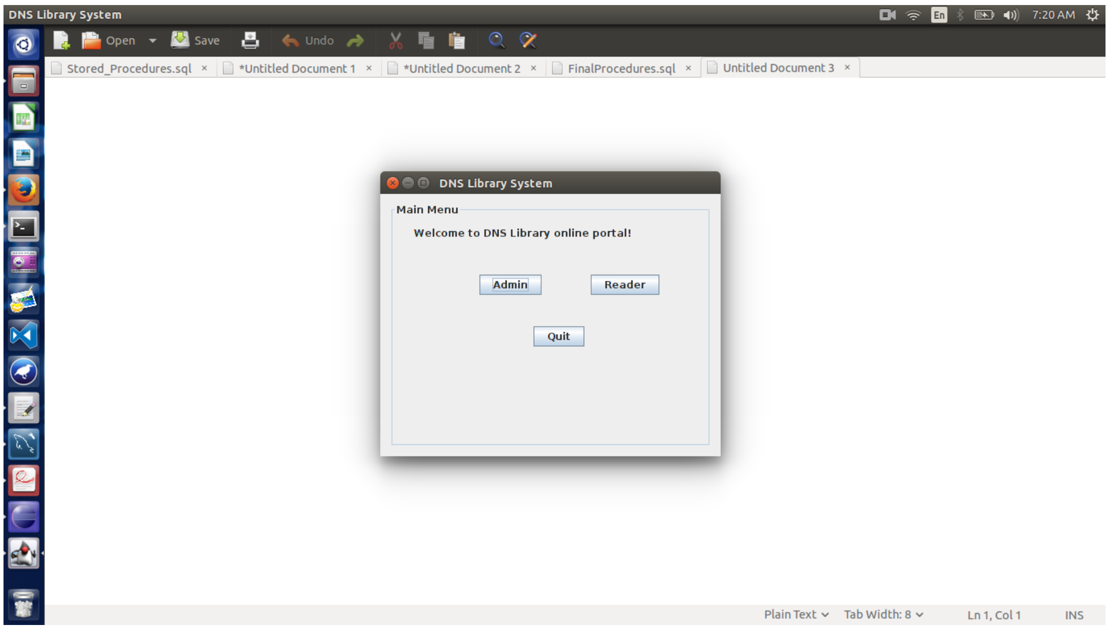


```
.CallableStatement;
okReturn {
urnTheBook(Connection connection, int
eStatement callableStatement = null;
bookReturnString = "{call book_return(
lableStatement = connection.prepareCall
lable
callabl
callabl
callabl
callabl
callabl
int active book = callableStatement.getInt
```

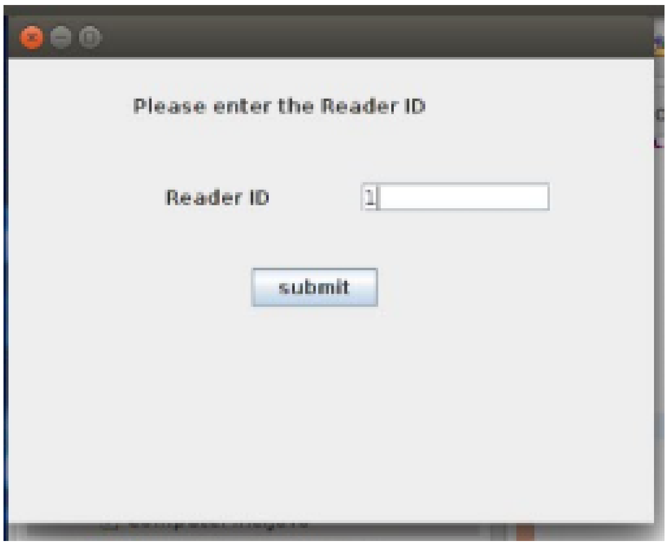
The screen above shows the average fine paid by the reader with reader id 1.

Reader Functionality

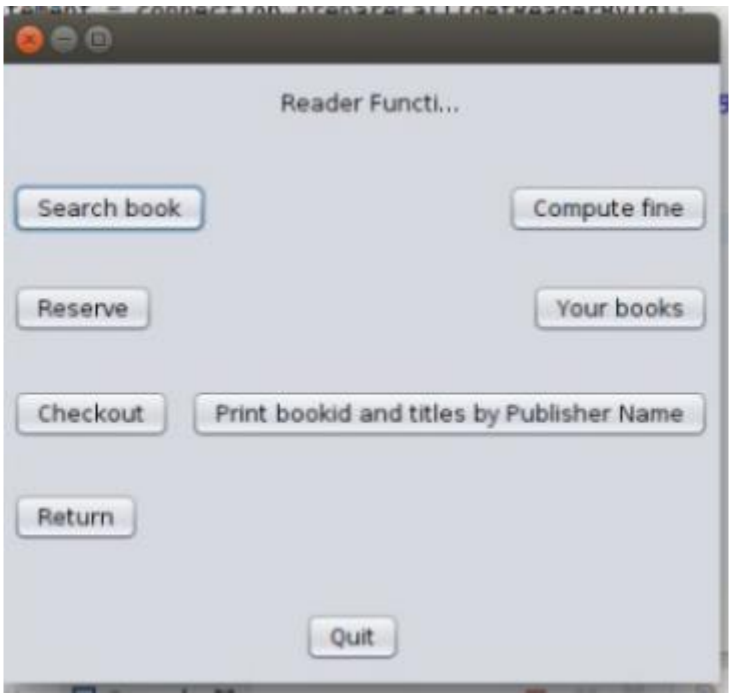
1. Reader Functions



Click on Reader to check the functionality:

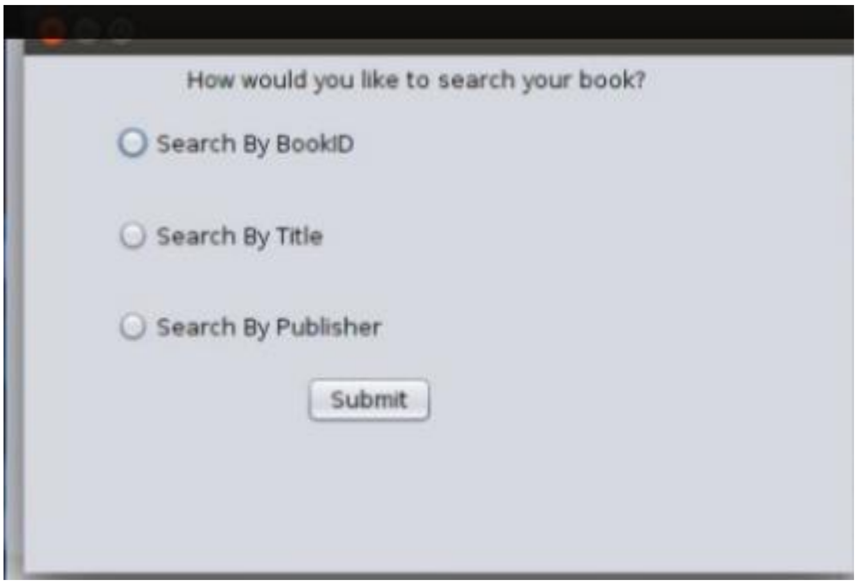


To login as a reader, window asks for the reader ID.

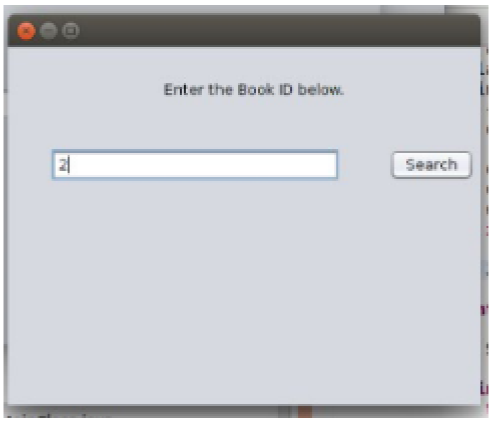


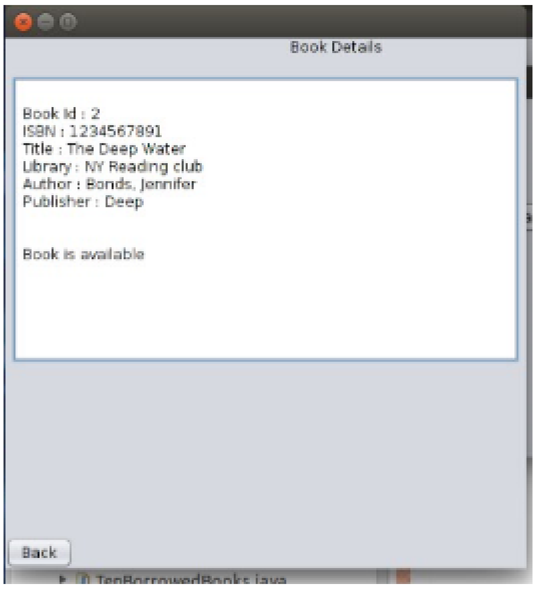
Above is the screen that reader sees after logging in.

2. **Search Options**



3. **Search Option 1 : Search by Book ID**

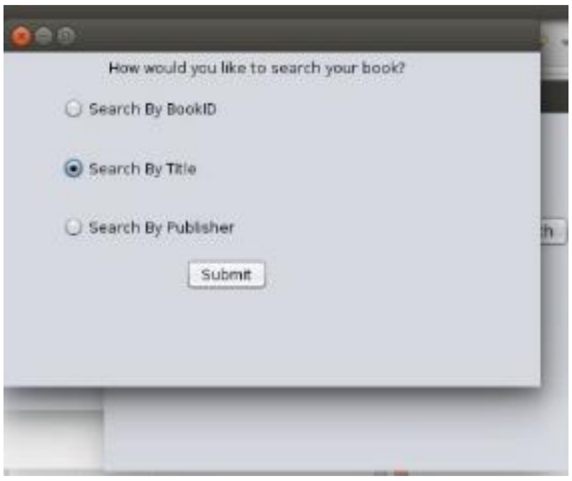




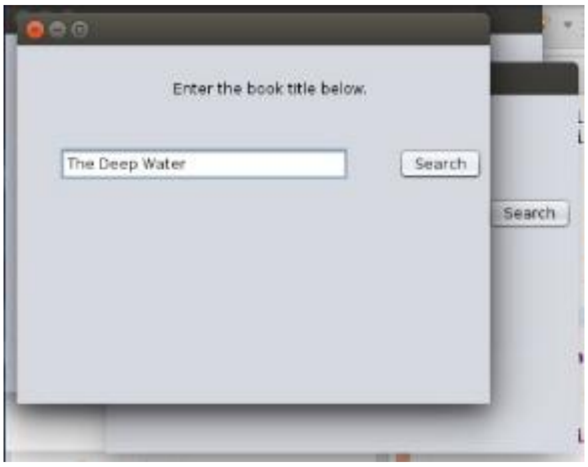
```
Book Id : 2
ISBN : 1234567891
Title : The Deep Water
Library : NY Reading club
Author : Bonds, Jennifer
Publisher : Deep


Book is available
```

Above is the screen that shows the result of search by bookID.

3. Search Option 2 : Search by Book title

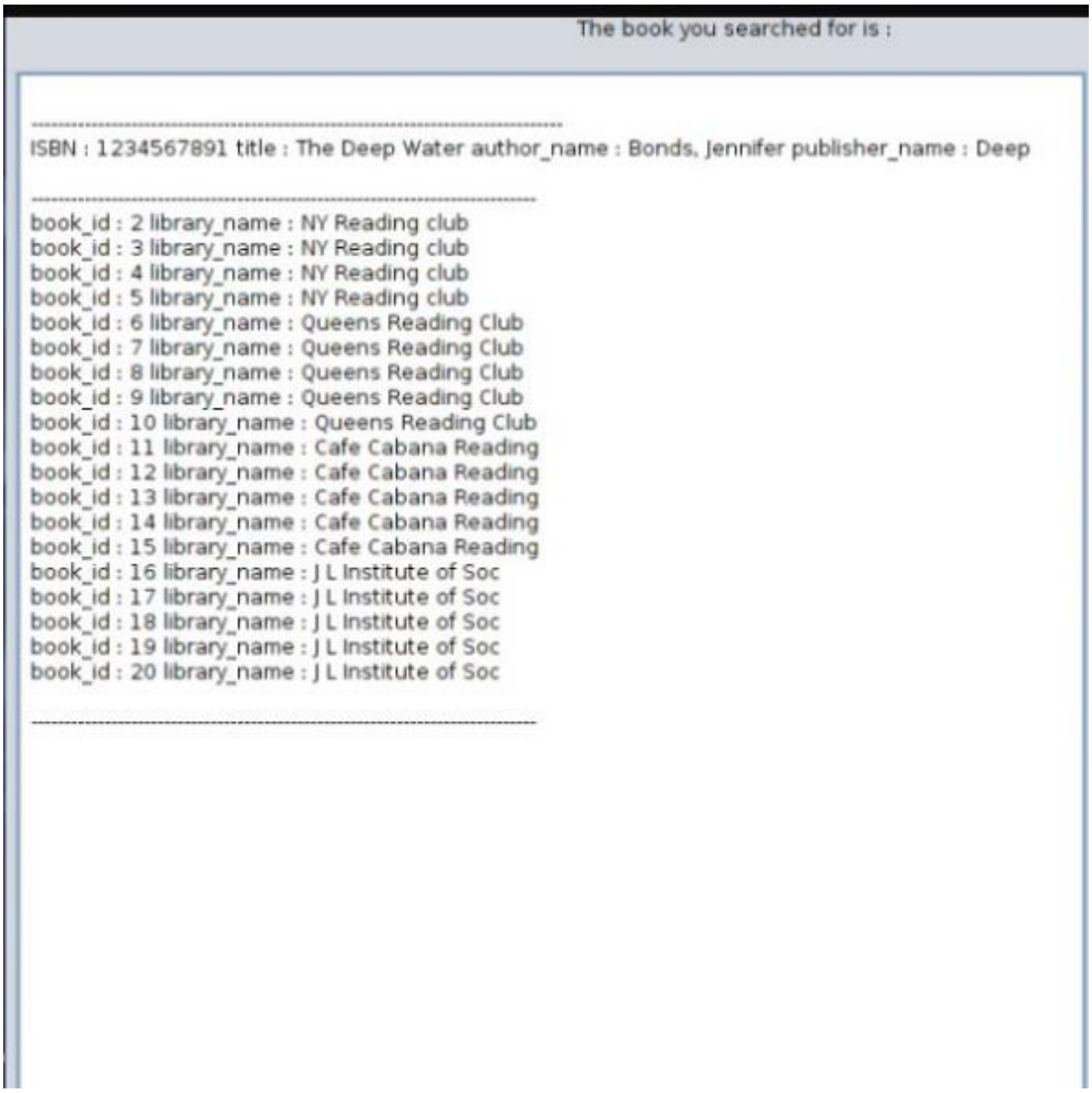


Search By BookID
[✓ Search By Title]
Search By Publisher



The Deep Water

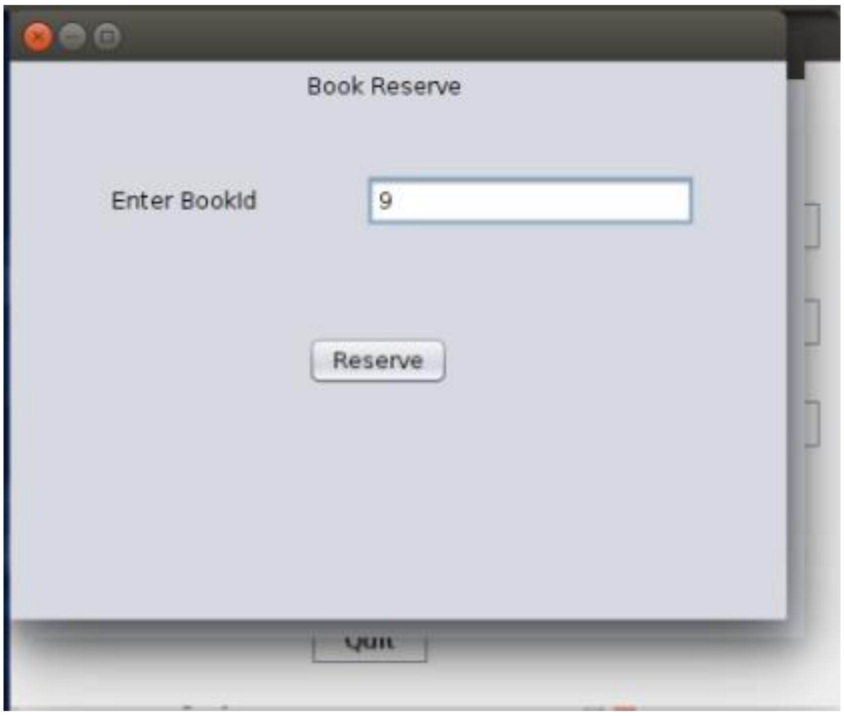
Above is the screen that shows the result of search by book title.



```
The book you searched for is :

------------------------------------------------------------------------
ISBN : 1234567891 title : The Deep Water author_name : Bonds, Jennifer publisher_name : Deep

--------------------------------------------------------------------
book_id : 2 library_name : NY Reading club
book_id : 3 library_name : NY Reading club
book_id : 4 library_name : NY Reading club
book_id : 5 library_name : NY Reading club
book_id : 6 library_name : Queens Reading Club
book_id : 7 library_name : Queens Reading Club
book_id : 8 library_name : Queens Reading Club
book_id : 9 library_name : Queens Reading Club
book_id : 10 library_name : Queens Reading Club
book_id : 11 library_name : Cafe Cabana Reading
book_id : 12 library_name : Cafe Cabana Reading
book_id : 13 library_name : Cafe Cabana Reading
book_id : 14 library_name : Cafe Cabana Reading
book_id : 15 library_name : Cafe Cabana Reading
book_id : 16 library_name : J L Institute of Soc
book_id : 17 library_name : J L Institute of Soc
book_id : 18 library_name : J L Institute of Soc
book_id : 19 library_name : J L Institute of Soc
book_id : 20 library_name : J L Institute of Soc

--------------------------------------------------------------------
```

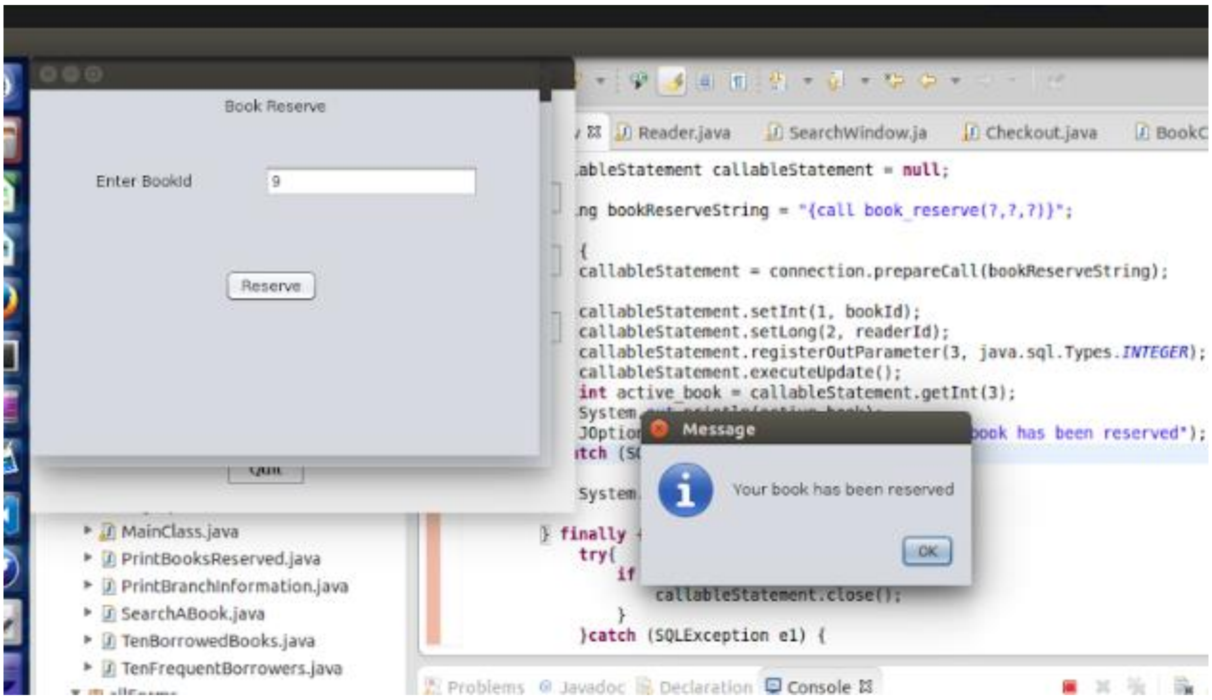
The screen above shows the search result and in which branches the book is available.

4. Reserve a book



Enter BookId: 9

The screen below shows the result after reserving a book.

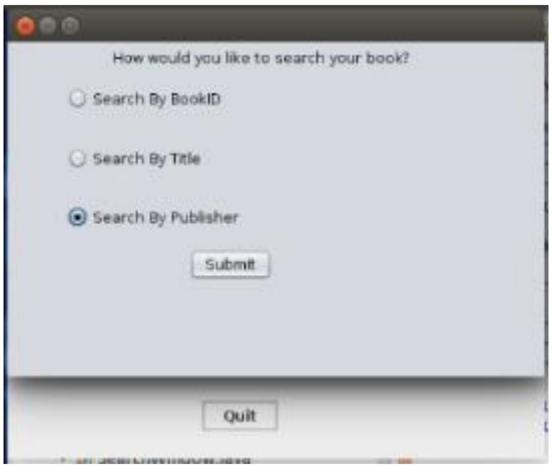


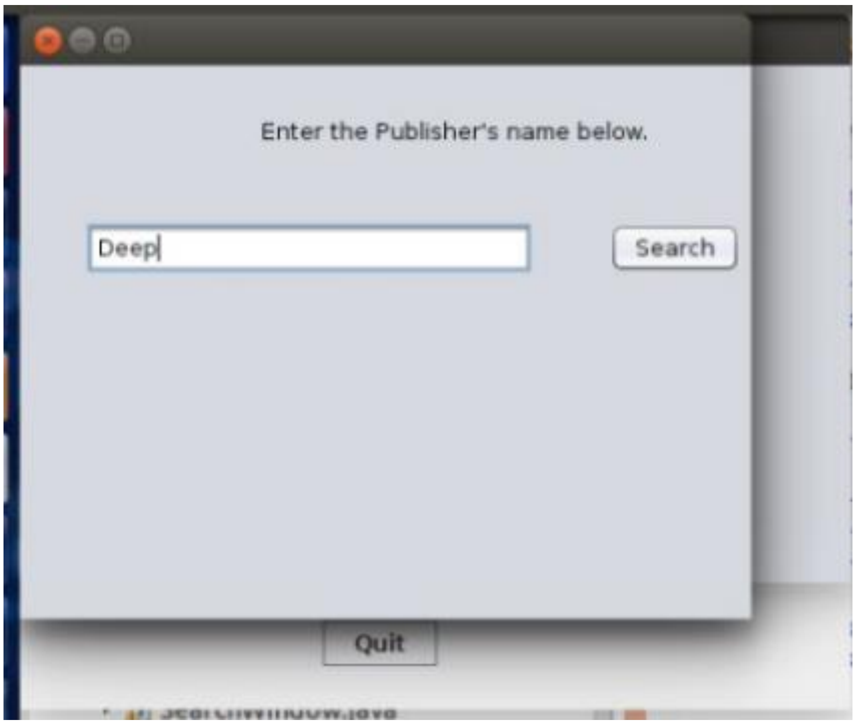
Enter BookId: 9

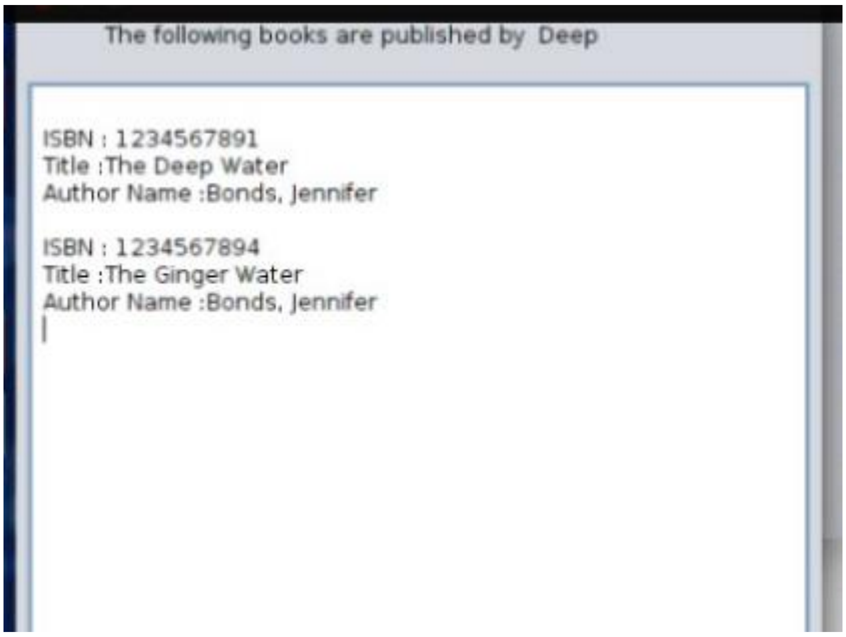
```
ableStatement callableStatement = null;

ng bookReserveString = "{call book_reserve(?,?,?)}";

{
callableStatement = connection.prepareCall(bookReserveString);

callableStatement.setInt(1, bookId);
callableStatement.setLong(2, readerId);
callableStatement.registerOutParameter(3, java.sql.Types.INTEGER);
callableStatement.executeUpdate();
int active_book = callableStatement.getInt(3);
System
JOption                                  book has been reserved");
tch (S

System

} finally
   try{
      if
         callableStatement.close();
      }
   }catch (SQLException e1) {
```

Your book has been reserved

5. Search Option 3 : Search by Publisher's Name



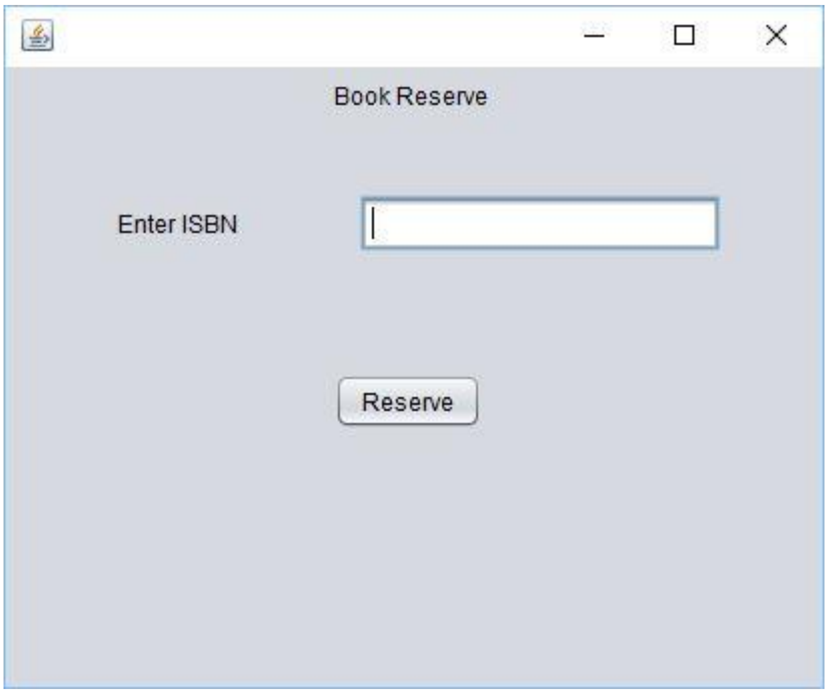




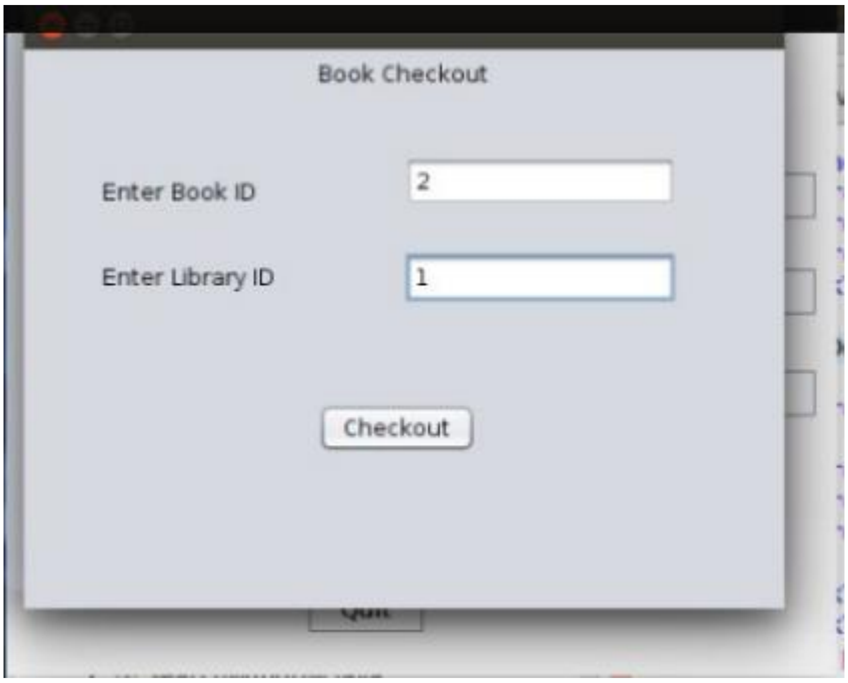
```
The following books are published by  Deep

ISBN : 1234567891
Title :The Deep Water
Author Name :Bonds, Jennifer

ISBN : 1234567894
Title :The Ginger Water
Author Name :Bonds, Jennifer
```

The screen above shows the result for the search of a publisher named deep. It shows the books published by this publisher.

6. Book Reserve



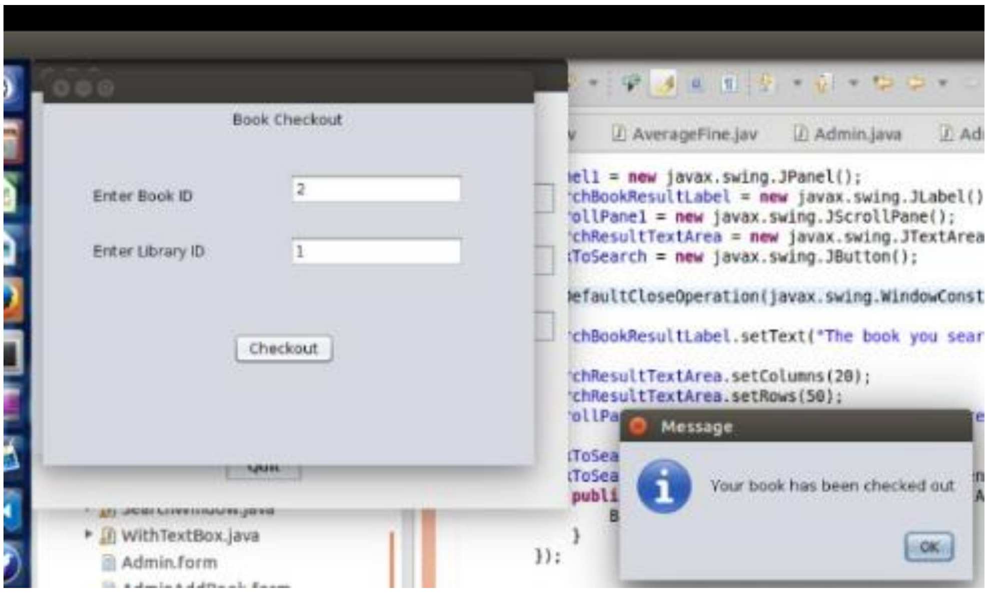
7. **Book Checkout**



Book Checkout

Enter Book ID: 2

Enter Library ID: 1



Book Checkout

Enter Book ID: 2

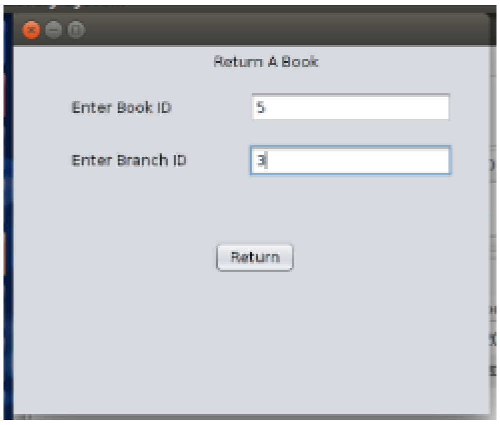
Enter Library ID: 1

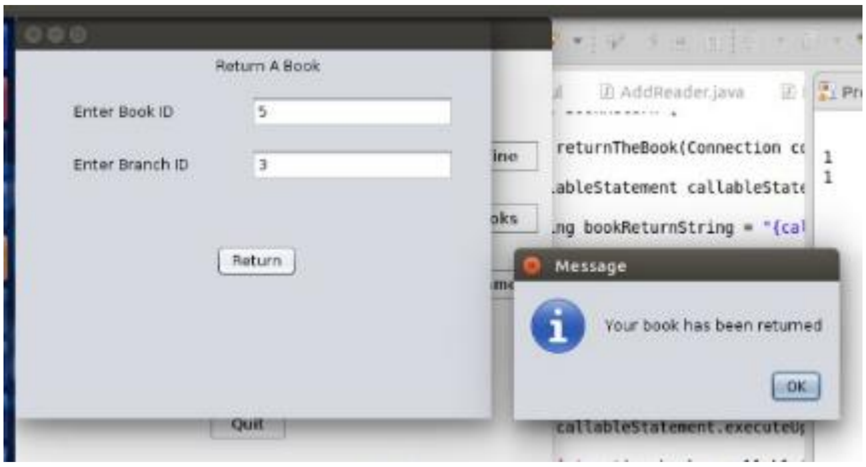
```
ell = new javax.swing.JPanel();
chBookResultLabel = new javax.swing.JLabel()
ollPanel = new javax.swing.JScrollPane();
chResultTextArea = new javax.swing.JTextArea
ToSearch = new javax.swing.JButton();

efaultCloseOperation(javax.swing.WindowConst

chBookResultLabel.setText("The book you sear

chResultTextArea.setColumns(20);
chResultTextArea.setRows(50);
```

Your book has been checked out

Screen above is the result after checking out with a book.

8. **Book Return**



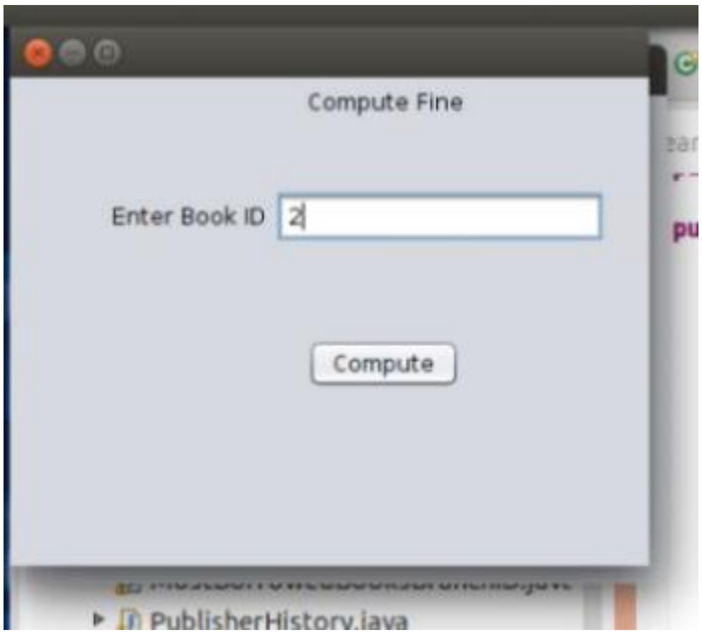


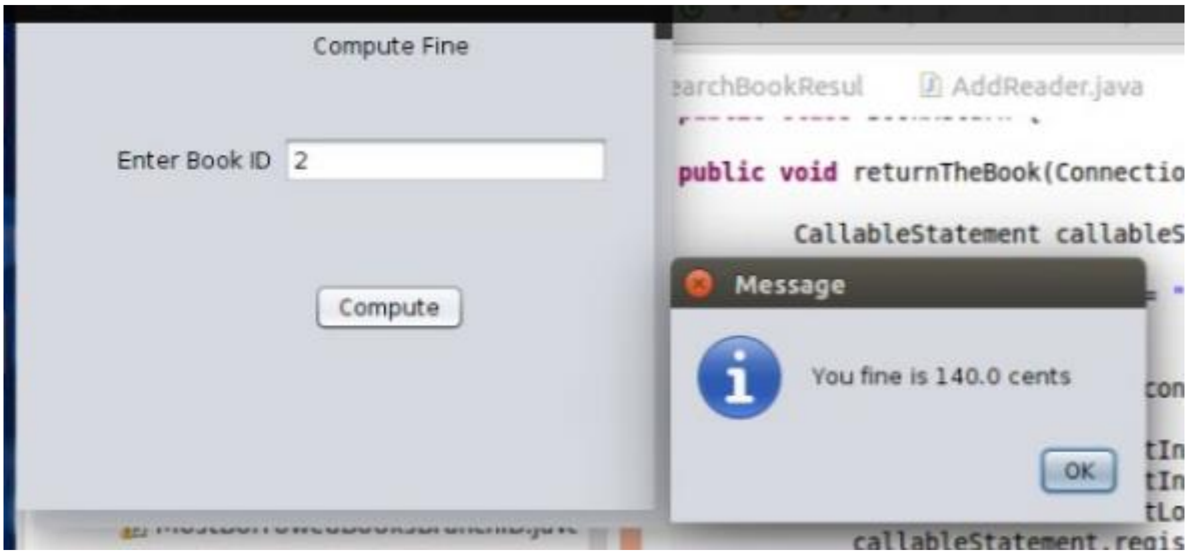
```
returnTheBook(Connection cc
.ableStatement callableState
.ng bookReturnString = "{cal
callableStatement.executeU
```

Your book has been returned

After returning a book the screen shows the result as shown in the screen above.

9. Compute Fine for Reader





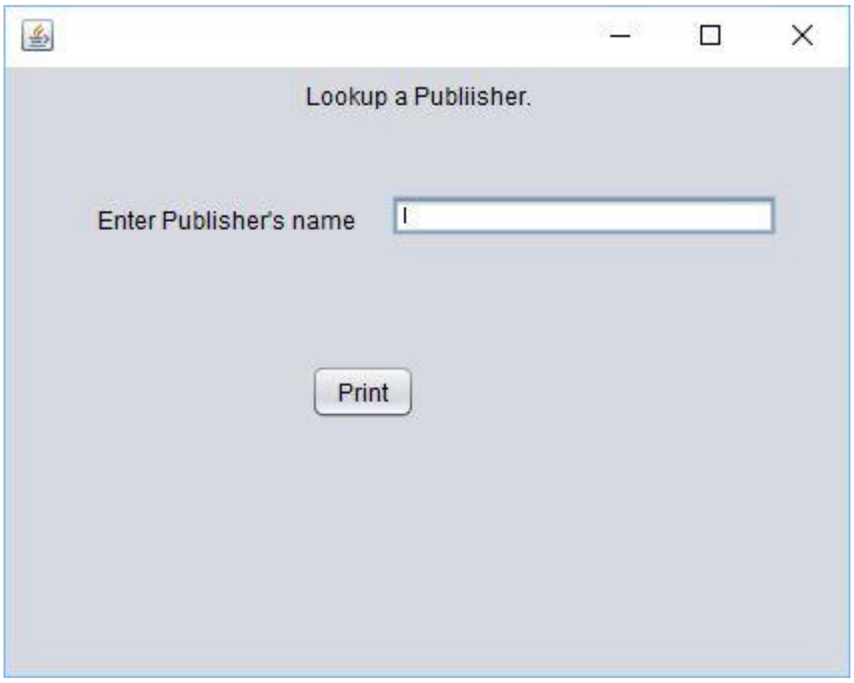
```
2
```

You fine is 140.0 cents

```
public void returnTheBook(Connectio
        CallableStatement callableS
callableStatement.regis
```

Above screen is the results after computing fine for a particular book for the reader that has logged into the system.

10. Look Up a Publisher


Lookup a Publiisher.
Enter Publisher's name
Print

6. Task Allocation

Shannon Mascarenhas (1076866)

- Programming Using Java Backend and implementation.
- Requirement Gathering - Programming Point of View
- Sending Requirement of Data Dependancy for Verification
- On Paper Flow Designing
- Store Procedure Developement
- Debugging and Testing the System

Niket Sagar (1060679)

- Database implementation on the localhost.
- UI Developement Using Java Swing
- Flow charts and analysis of the implementation.
- Report and Documentation
- Logic Discussion with Shannon for the Back End Logic
- Debugging and Testing the System

Deep Desai (1059666)

- Requirement gathering and analysis of the system.
- Requirement gathering and analysis of the database.
- Database design and creating testing dataset
- SQL query building and Store Procedure Modification
- Report and documentation.
- Data Entries Using Import Function in the MySql