CS 631 - DATA MANAGEMENT SYSTEMS DESIGN

SECTION 001
PROJECT
The City Library

Deliverable 3

Description of Implementation

Submitted by

Sunny Patel
31420957
smp222@njit.edu

Sanket Shankar Kulkarni
31406022
sk2339@njit.edu

Harshil Shah
31434597
hns8@njit.edu

Description of Implementation

1. **Database:**
    - The database used in this application is MSSQL.
    - There are total 17 tables and 7 stored procedures used to implement the database.
    - All 17 tables are designed (no. of columns, Primary Key Constraints, Foreign Key Constraints) according to “Relational Schema” diagram created in Deliverable 2, except one change. A new column named “Reservation_Status” is added in table “Reservation” to avoid several long and repetitive computations. (Updated diagram is attached in this document.)
    - There are 7 stored procedures designed to do some complex computations before providing data to the user.
    - Brief description and additional constraints of all **Tables:**
        a. **Branch:** This table contains details of all branches: Id, Name and Location. All fields are required.
        b. **Reader:** This table contains details of all registered readers: Id, Name, Type, Phone no and Address. All fields are required.
        c. **Publisher:** This table contains details of all publishers: Id, Name and Address. All fields are required.
        d. **Bor_Transaction:** This table contains records of Borrow Time and Return Time for a “Borrow” transaction. The entries having Return Time field null, are not returned yet.
        e. **Reservation:** This table contains records of Reservation Date and Reservation Status for a “Reserve” transaction.
        f. **Chief_editor:** This table contains information of chief editors of journals: Id and Name. All fields are required.
        g. **Author:** This table contains information of authors of books: Id and Name. All fields are required.
        h. **Document:** This table contains information of all the documents: Id, Title, Published date, Publisher Id. The title cannot be null.
        i. **Book:** This table contains information of all the books: Id, ISBN number. All fields are required.
        j. **Writes:** This table contains mapping of authors and books they have written: Document Id, Author Id. All fields are required.
        k. **Proceedings:** This table contains details of all conference proceedings: Id, Conference Date, Conference location, Conference editor. Conference date is required field.
        l. **Journal_Volume:** This table contains information regarding journal volumes: Id, Volume no., Editor Id. All fields are required.
        m. **Journal_Issue:** This table contains information of Journal Issues: Id, Issue no., Scope.
        n. **Guest_Editors:** This table contains information regarding guest editors: Id, Issue no, Name.

o. **Copy:** This table contains information of copies of all documents: Library Id, Document Id, Copy no, Position. All fields are required.

p. **Reserves:** This table contains all information of a “Reserve” transaction: Document Id, Library Id, Copy no, Reader Id. All fields are required.

q. **Borrows:** This table contains all information of a “Borrow” transaction: Document Id, Library Id, Copy no, Reader Id. All fields are required.

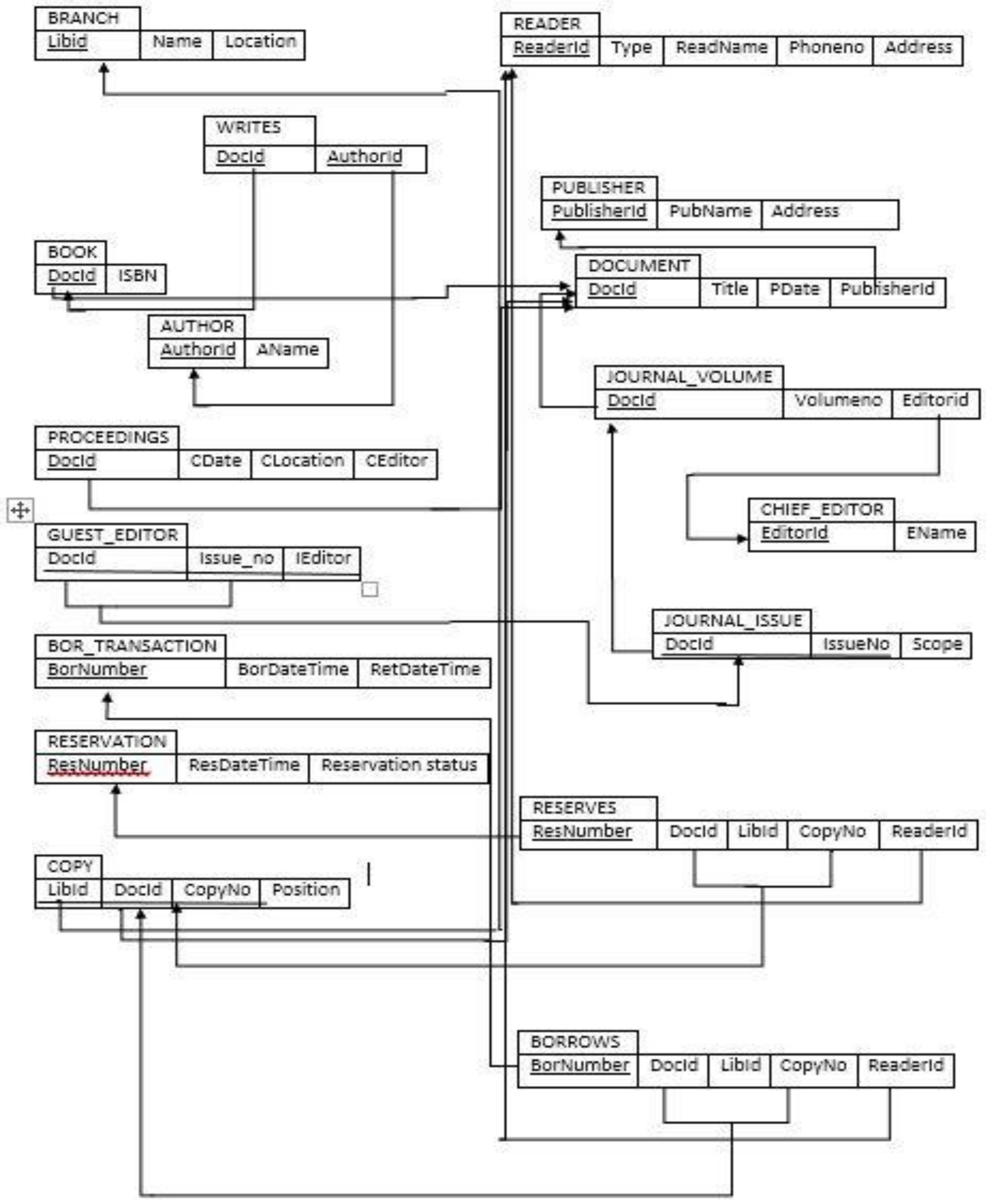
- Brief descriptions of all **Stored Procedures**:
  - a. **spAddDocument:** This stored procedure helps to add document if valid details are provided
    - i. Input parameters: Document title, Published Date, Publisher’s id, Library id, Copy no, Position.
    - ii. Output Parameter: Status
    - iii. Functions:
      1. Checks if given publisher and branch are present.
      2. Generates document id and Inserts data into Document and Copy tables.
  - b. **spAddReader:** This stored procedure helps to add a new reader in system.
    - i. Input parameters: Reader name, Type, Phone no, address
    - ii. Output Parameter: NONE
    - iii. Functions:
      1. Generates Reader id and inserts data into Reader table.
  - c. **spCheckDocStatus:** This stored procedure fetches current status of a document copy.
    - i. Input parameters: Document Id, Library id, Copy no.
    - ii. Output Parameter: Status
    - iii. Functions:
      1. Updates Reservation table and sets status of documents to ‘Cancelled’ if they were not checked out before 6:00 pm on their reservation date.
      2. Checks if document is present, borrowed or reserved.
      3. Correct status is reflected in output parameter.
  - d. **spCheckout:** This stored procedure checks out borrowed documents.
    - i. Input parameters: Reader Id, Document Id, Library id, Copy no.
    - ii. Output Parameter: Result
    - iii. Functions:
      1. Updates Reservation table and sets status of documents to ‘Cancelled’ if they were not checked out before 6:00 pm on their reservation date.
      2. Checks if document is present, borrowed or reserved by another user.
      3. Generates transaction number, inserts data into Borrows and Bor_Transaction tables.

4. Updates Reservation table if the document was reserved by current user.
5. Sets appropriate result

e. **spGetReservedDocs:** This stored procedure returns the details of all reserved documents by current user.
   i. Input parameters: Reader Id
   ii. Output Parameter: NONE
   iii. Functions:
      1. Updates Reservation table and sets status of documents to ‘Cancelled’ if they were not checked out before 6:00 pm on their reservation date.
      2. Returns details of all reserved documents by current user.

f. **spReserve:** This stored procedure reserves a document for current user.
   i. Input parameters: Reader Id, Document Id, Library id, Copy no, Reservation Date.
   ii. Output Parameter: Result
   iii. Functions:
      1. Updates Reservation table and sets status of documents to ‘Cancelled’ if they were not checked out before 6:00 pm on their reservation date.
      2. Checks if a document is borrowed or reserved.
      3. Generates reservation number, inserts data into Reservation and Reserves tables.
      4. Sets appropriate value for result parameter.

g. **spReturn:** This stored procedure handles event of returning a document.
   i. Input parameters: Reader Id, Document Id, Library id, Copy no.
   ii. Output Parameter: Result
   iii. Functions:
      1. Checks if the document was borrowed.
      2. Updates table Bor_Transaction and records return time.
      3. Sets appropriate value for Result.


BRANCH
Libid
Name
Location
READER
ReaderId
Type
ReadName
Phoneno
Address
WRITES
DocId
AuthorId
PUBLISHER
PublisherId
PubName
Address
BOOK
DocId
ISBN
DOCUMENT
DocId
Title
PDate
PublisherId
AUTHOR
AuthorId
AName
JOURNAL_VOLUME
DocId
Volumeno
EditorId
PROCEEDINGS
DocId
CDate
CLocation
CEditor
CHIEF_EDITOR
EditorId
EName
GUEST_EDITOR
DocId
Issue_no
IEditor
JOURNAL_ISSUE
DocId
IssueNo
Scope
BOR_TRANSACTION
BorNumber
BorDateTime
RetDateTime
RESERVATION
ResNumber
ResDateTime
Reservation status
RESERVES
ResNumber
DocId
LibId
CopyNo
ReaderId
COPY
LibId
DocId
CopyNo
Position
BORROWS
BorNumber
DocId
LibId
CopyNo
ReaderId

2. <u>Application</u>:

- The front end of this application is developed using ASP.NET MVC 5.
- In MVC, whole application is divided in 3 major components: Model, View and Controller (MVC).
    - **a. <u>Model</u>:** It handles database operations.
    - **b. <u>View</u>:** It handles how data is represented to the user.
    - **c. <u>Controller</u>:** It works as middle layer between Model and View. It determines which View should be displayed to user on a action performed. It is also responsible for transferring data obtained from model to a View.
- ADO.NET core is used to connect application to database.
- The application has two models:
    - **a. CityLibraryDAL:** It is responsible for fetching data for User mode functions.
    - **b. CityLibraryAdminDAL:** It is responsible for fetching data for Admin mode functions.
- The application has three controllers:
    - **a. HomeController:** It handles events on homepage of application.
    - **b. AdminController:** It handles events of Admin functions.
    - **c. UserController:** It handles events of User functions.
- For each controller, there is one folder in Views folder. It contains pages related to that controller. Thus, there are 3 folders in View: admin, user and home. In admin, there are 14 views, in User there are 12 views and for home there is 1 view. These views are responsible to display content to application's user.
- In user mode, application first requires user to enter reader id. If entered correct id, user's reader id is saved and is used throughout the application i. e. Any data displayed to the user will be related to the current user. The user won't be asked again for his/her id or name anywhere in application. These values will be passed implicitly. When user selects "Quit" option on main menu, these details are removed, and user gets logged out. Also on each page of user mode, it is checked that system has data of a user who has logged in, if accessed directly, these pages will auto redirect to homepage of the application.
- For admin, id and passwords are stored in "web.config" file. **We have assumed that there is only one admin,** so it is not rational to store his/her details in database. Whenever admin enters credentials, systems saves its data and keeps it till admin clicks on "Quit" option on the main menu. Just like pages in user domain, admin pages also check if admin's data is available in system when they are accessed. If not, they redirect to homepage of the application.
- All other functions or details are according to provided guidelines.

3. <u>Problems Faced</u>:

- We faced issue for implementing a requirement:" The reserved book should be borrowed before or on reserved date before 6:00 pm." As it is an action which is triggered based on a time event, we can't figure out how to implement it. We have now implemented it like whenever a query is made from application to view status of document or to reserve or borrow a document, we check all reserved items against the current time stamp. If their reservation time is older than 6:00 pm on current date, we cancel their registration.
- Thus, even if a document's (whose reservation date has been passed) status is reserved in database, whenever an attempt is made to do anyone of the above three operations, we correct its status.
- As there is no other way to see the status for user or admin, this solution, although not sophisticated, will work in any case.

4. Source Code:

- **Create queries:**
    - CREATE TABLE Branch (Libid INT, Name nvarchar(50) NOT NULL, Location nvarchar(50) NOT NULL, CONSTRAINT pk_Branch PRIMARY KEY (Libid))

    - CREATE TABLE Reader (Reader_id INT, Type nvarchar(20) NOT NULL, ReaderName nvarchar(50) NOT NULL, Phone_no nvarchar(12) NOT NULL, Address nvarchar(100) NOT NULL, CONSTRAINT pk_Reader PRIMARY KEY (Reader_id), CONSTRAINT uq_Reader_Phone UNIQUE (Phone_no))

    - CREATE TABLE Publisher (Publisher_id INT, Pub_Name nvarchar(50) NOT NULL, Address nvarchar(100) NOT NULL, CONSTRAINT pk_Publisher PRIMARY KEY (Publisher_id), CONSTRAINT uq_PubName UNIQUE(Pub_Name))

    - CREATE TABLE BOR_Transaction (BorNumber INT, BorDateTime DATETIME NOT NULL, RetDateTime DATETIME, CONSTRAINT pk_BOR_Transaction Primary key (BorNumber), CONSTRAINT chk_bor_rettime CHECK(BorDateTime < RetDateTime))

    - CREATE TABLE Reservation (ResNumber int, ResDateTime datetime NOT NULL, Reservation_Status nvarchar(10) constraint pk_Reservation PRIMARY key (ResNumber))

    - CREATE TABLE Chief_Editor (EditorId int, Ename nvarchar(50) NOT NULL, CONSTRAINT pk_Cheif_Editor primary key (EditorId))

    - CREATE TABLE Author (AuthorId int, Aname nvarchar(50) NOT NULL, constraint pk_Author primary key (AuthorId))

    - CREATE TABLE Document (DocId int, Title nvarchar(50) NOT NULL, Pdate DATE, PublisherID int, CONSTRAINT pk_Document Primary key (DocId), Constraint fk_Document_Publisher Foreign key (PublisherId) references Publisher)

    - CREATE TABLE Book (DocId int, ISBN nvarchar(15) NOT NULL, constraint pk_Book primary key (DocId), Constraint fk_Book_Document FOREIGN KEY (DocId) REFERENCES Document)
    - CREATE TABLE Writes (DocId int, AuthorId int, constraint pk_Writes Primary Key (DocId,AuthorId), Constraint fk_Writes_Book FOREIGN KEY (DocId) References Book, Constraint fk_Writes_Author FOREIGN KEY (AuthorId) References Author)

- CREATE TABLE Proceedings (DocId INT, CDate DATE NOT NULL, Clocation nvarchar(50), Ceditor nvarchar(50) NOT NULL, constraint pk_Proceedings Primary Key (DocId), Constraint fk_Proceedings_Document FOREIGN KEY (DocId) References Document)

- CREATE TABLE Journal_Volume (DocId INT, Volumeno int NOT NULL, Editorid INT NOT NULL, Constraint pk_Journal_Volume Primary key (DocId), Constraint fk_Journal_Volume_Chief_Editor FOREIGN KEY (Editorid) References Chief_Editor, Constraint fk_Journal_Volume_Document FOREIGN KEY (DocId) References Document)

- CREATE TABLE Journal_Issue (DocId int, Issueno int, Scope nvarchar(10), Constraint pk_Journal_Issue Primary Key (DOcId,Issueno), Constraint fk_Journal_Issue_Journal_Volume FOREIGN KEY (DocId) References Journal_Volume)

- CREATE TABLE Guest_Editor (DocId INT, Issueno INT, IEditor nvarchar(50), Constraint pk_Guest_Editor Primary Key (DocId, Issueno, IEditor), Constraint fk_Guest_Editor_Journal_Issue FOREIGN KEY (DocId,Issueno) References Journal_Issue)

- CREATE TABLE Copy(LibId int, DocID int, Copyno int, Position char(6) NOT NULL, Constraint pk_Copy Primary key (LibId, DocId, Copyno), Constraint fk_Copy_Branch FOREIGN KEY (LibId) References Branch,Constraint fk_Copy_Document FOREIGN KEY (DocId) References Document)

- Create Table Reserves(ResNumber int,DocID int NOT NULL, LibId int NOT NULL, Copyno int NOT NULL,ReaderId int NOT NULL, Constraint pk_Reserves primary key (ResNumber),Constraint fk_Reserves_reader FOREIGN KEY (ReaderId) References Reader,Constraint fk_Reserves_Copy FOREIGN KEY (LibId, DocId, Copyno) References Copy,Constraint fk_Reserves_Reservation FOREIGN KEY (ResNumber) References Reservation)

- Create Table Borrows(BorNumber int,DocID int NOT NULL, LibId int NOT NULL, Copyno int NOT NULL,ReaderId int NOT NULL, Constraint pk_Borrows primary key (BorNumber),Constraint fk_Borrows_reader FOREIGN KEY (ReaderId) References Reader,Constraint fk_Borrows_Copy FOREIGN KEY (LibId, DocId, Copyno) References Copy,Constraint fk_Borrows_BOR_Transaction FOREIGN KEY (BorNumber) References Bor_Transaction)

- Insert queries:
  - -- Branch
  - INSERT INTO Branch VALUES (1,'New Port','Jersey City')
  - INSERT INTO Branch VALUES (2,'Newark','Newark')

- INSERT INTO Branch VALUES (3,'Chatham','Chatham')
- INSERT INTO Branch VALUES (4,'Pennington','Pennington')
- INSERT INTO Branch VALUES (5,'Lincroft','Lincroft')
- INSERT INTO Branch VALUES (6,'Lambertville','Lambertville')
- INSERT INTO Branch VALUES (7,'River Edge','River Edge')
- INSERT INTO Branch VALUES (8,'Ho-Ho-Kus','Ho-Ho-Kus')
- INSERT INTO Branch VALUES (9,'Little Silver','Little Silver')
- INSERT INTO Branch VALUES (10,'New Providence','New Providence')
- INSERT INTO Branch VALUES (11,'Robbinsville','Robbinsville')
- INSERT INTO Branch VALUES (12,'Haddonfield','Haddonfield')
- INSERT INTO Branch VALUES (13,'Medford Lakes','Medford Lakes')
- INSERT INTO Branch VALUES (14,'Clinton','Clinton')
- INSERT INTO Branch VALUES (15,'Dumont','Dumont')
- INSERT INTO Branch VALUES (16,'Tenafly','Tenafly')
- INSERT INTO Branch VALUES (17,'Rumson','Rumson')
- INSERT INTO Branch VALUES (18,'Waldwick','Waldwick')
- INSERT INTO Branch VALUES (19,'Heathcote','Heathcote')
- INSERT INTO Branch VALUES (20,'Ramsey','Ramsey')
- -- Reader
-
- INSERT INTO Reader VALUES(1001,'Student','Kaycee Deyo','+14898724788','8577 Lakewood St. Addison, IL 60101')
- INSERT INTO Reader VALUES(1002,'Senior citizen','Frederica Burgher','+13783776848','208 Catherine Ave. Greer, SC 29650')
- INSERT INTO Reader VALUES(1003,'Senior citizen','Roni Polston','+14292624943','91 Lakeview St. Saginaw, MI 48601')
- INSERT INTO Reader VALUES(1004,'Student','Shane Barney','+18964232974','837 Old Ridge St. West Orange, NJ 07052')
- INSERT INTO Reader VALUES(1005,'Staff','Marisa Stamant','+17984848224','9099 Trout Lane San Diego, CA 92111')
- INSERT INTO Reader VALUES(1006,'Staff','Meagan Cornforth','+17683249992','882 John Ave. Atlantic City, NJ 08401')
- INSERT INTO Reader VALUES(1007,'Student','Eulalia Pletcher','+18893836872','636 South Mayflower St. Redondo Beach, CA 90278')
- INSERT INTO Reader VALUES(1008,'Senior citizen','Bert Pawloski','+17889398764','84 Marsh Ave. Orange Park, FL 32065')
- INSERT INTO Reader VALUES(1009,'Senior citizen','Waneta Taniguchi','+13693979289','9028 N. Shore St. Huntersville, NC 28078')
- INSERT INTO Reader VALUES(1010,'Student','Zachariah Brault','+19836886494','681 Atlantic Court Massapequa, NY 11758')
- INSERT INTO Reader VALUES(1011,'Staff','Roberto Westray','+13799779883','920 East Leeton Ridge St. Ogden, UT 84404')
- INSERT INTO Reader VALUES(1012,'Staff','Cody Nocera','+18288429788','85 Inverness Dr. Louisville, KY 40207')

- INSERT INTO Reader VALUES(1013,'Student','Darryl Chatfield','+18797282463','24 Mayfield Ave. Newington, CT 06111')
- INSERT INTO Reader VALUES(1014,'Senior citizen','Kia Bialek','+16337223396','3 Broad Court Granger, IN 46530')
- INSERT INTO Reader VALUES(1015,'Staff','Sabra Raygoza','+17434486992','368 Wild Horse Lane North Wales, PA 19454')
- INSERT INTO Reader VALUES(1016,'Student','Sheba Rivero','+19624682636','7610 North Tarkiln Hill St. Coachella, CA 92236')
- INSERT INTO Reader VALUES(1017,'Staff','Alla Collette','+12894833488','230 Winding Way Street Dearborn, MI 4812')
- INSERT INTO Reader VALUES(1018,'Staff','Clorinda Goewey','+19964269996','97 Green Lake Avenue Jupiter, FL 33458')
- INSERT INTO Reader VALUES(1019,'Student','Dania Ouk','+14427774679','97 Green Lake Avenue Jupiter, FL 33458')
- INSERT INTO Reader VALUES(1020,'Senior citizen','Carolyn Jessie','+17428887322','144 Whitemarsh Avenue Greensburg, PA 15601')
-
- --publisher
-
- INSERT INTO Publisher VALUES(2001,'Kati Compo','647 Baverian Hills, LA 03678')
- INSERT INTO Publisher VALUES(2002,'Woodrow Ham','9594 New Saddle St. Watertown, MA 02472')
- INSERT INTO Publisher VALUES(2003,'Alexandria Beadles','8814 Forest St. Rock Hill, SC 29730')
- INSERT INTO Publisher VALUES(2004,'Augustina Malm','183 Mammoth Ave. Buford, GA 30518')
- INSERT INTO Publisher VALUES(2005,'Ewa Grosvenor','13 Lantern Ave. Phoenixville, PA 19460')
- INSERT INTO Publisher VALUES(2006,'Henrietta Desir','9791 Harvey Avenue Maumee, OH 43537')
- INSERT INTO Publisher VALUES(2007,'Kathaleen Vaugh','418 Carriage Lane Southampton, PA 18966')
- INSERT INTO Publisher VALUES(2008,'Alethea Rentfro','29 Shore Court Holly Springs, NC 27540')
- INSERT INTO Publisher VALUES(2009,'See Corley','34 Nut Swamp Street Dorchester, MA 02125')
- INSERT INTO Publisher VALUES(2010,'Lucio Whitton','95 N. Pine Dr. Conway, SC 29526')
- INSERT INTO Publisher VALUES(2011,'Clarissa Fabry','4 East Purple Finch Drive Huntington, NY 11743')
- INSERT INTO Publisher VALUES(2012,'Angelica Chickering','8898 Rockwell Lane Apopka, FL 32703')

- INSERT INTO Publisher VALUES(2013,'Adolph Canon','7769 Mayfair Street Martinsville, VA 24112')
- INSERT INTO Publisher VALUES(2014,'Lessie Grimaldo','101 Middle River St. Spring Valley, NY 10977')
- INSERT INTO Publisher VALUES(2015,'Arleen Manahan','7294 Shady Ave. Palm Coast, FL 32137')
- INSERT INTO Publisher VALUES(2016,'Coleman Langsam','855 Bear Hill Court Freehold, NJ 07728')
- INSERT INTO Publisher VALUES(2017,'Joleen Koren','8524 San Juan St. Deltona, FL 32725')
- INSERT INTO Publisher VALUES(2018,'Asia Riker','4 Ketch Harbour Lane Olympia, WA 98512')
- INSERT INTO Publisher VALUES(2019,'Courtney Sinclair','7913 Country Club Street Wappingers Falls, NY 12590')
- INSERT INTO Publisher VALUES(2020,'Tandra Kantor','667 Birchwood Ave. Wyandotte, MI 48192')
-
- --BOR_Transaction
- INSERT INTO BOR_Transaction VALUES(3001,'1-3-2017 09:20','2-13-2017 09:20')
- INSERT INTO BOR_Transaction VALUES(3002,'1-13-2017 17:00','2-20-2017 17:00')
- INSERT INTO BOR_Transaction VALUES(3003,'1-23-2017 16:40','2-21-2017 16:40')
- INSERT INTO BOR_Transaction VALUES(3004,'1-24-2017 16:20','5-16-2017 16:20')
- INSERT INTO BOR_Transaction VALUES(3005,'2-2-2017 16:00','5-23-2017 16:00')
- INSERT INTO BOR_Transaction VALUES(3006,'2-9-2017 15:40','5-25-2017 15:40')
- INSERT INTO BOR_Transaction VALUES(3007,'3-6-2017 15:20','6-12-2017 15:20')
- INSERT INTO BOR_Transaction VALUES(3008,'3-24-2017 15:00','6-16-2017 15:00')
- INSERT INTO BOR_Transaction VALUES(3009,'4-6-2017 14:20','6-19-2017 14:20')
- INSERT INTO BOR_Transaction VALUES(3010,'5-11-2017 13:40','6-30-2017 13:40')
- INSERT INTO BOR_Transaction VALUES(3011,'6-14-2017 13:20','7-18-2017 13:20')
- INSERT INTO BOR_Transaction VALUES(3012,'7-24-2017 13:00','7-28-2017 13:00')
- INSERT INTO BOR_Transaction VALUES(3013,'8-23-2017 12:40','9-8-2017 12:40')

- INSERT INTO BOR_Transaction VALUES(3014,'9-13-2017 12:20','9-19-2017 12:20')
- INSERT INTO BOR_Transaction VALUES(3015,'9-15-2017 12:00','9-25-2017 12:00')
- INSERT INTO BOR_Transaction VALUES(3016,'9-25-2017 11:40','10-13-2017 11:40')
- INSERT INTO BOR_Transaction VALUES(3017,'9-25-2017 11:20','10-23-2017 11:20')
- INSERT INTO BOR_Transaction VALUES(3018,'11-20-2017 10:40','11-21-2017 10:40')
- INSERT INTO BOR_Transaction VALUES(3019,'11-21-2017 10:20','12-20-2017 10:20')
- INSERT INTO BOR_Transaction VALUES(3020,'11-21-2017 10:00','12-27-2017 10:00')
- INSERT INTO BOR_Transaction VALUES(3021,'2017-01-03 09:20:00.000','2017-02-13 09:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3022,'2017-01-13 17:00:00.000','2017-02-20 17:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3023,'2017-01-23 16:40:00.000','2017-02-21 16:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3024,'2017-01-24 16:20:00.000','2017-05-16 16:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3025,'2017-02-02 16:00:00.000','2017-05-23 16:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3026,'2017-02-09 15:40:00.000','2017-05-25 15:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3027,'2017-03-06 15:20:00.000','2017-06-12 15:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3028,'2017-03-24 15:00:00.000','2017-06-16 15:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3029,'2017-04-06 14:20:00.000','2017-06-19 14:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3030,'2017-05-11 13:40:00.000','2017-06-30 13:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3031,'2017-06-14 13:20:00.000','2017-07-18 13:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3032,'7-24-2017 13:00','7-28-2017 13:00')
- INSERT INTO BOR_Transaction VALUES(3033,'8-23-2017 12:40','9-8-2017 12:40')
- INSERT INTO BOR_Transaction VALUES(3034,'9-13-2017 12:20','9-19-2017 12:20')
- INSERT INTO BOR_Transaction VALUES(3035,'9-15-2017 12:00','9-25-2017 12:00')

- INSERT INTO BOR_Transaction VALUES(3036,'2017-09-25 11:40:00.000','2017-10-13 11:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3037,'2017-09-25 11:20:00.000','2017-10-23 11:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3038,'11-20-2017 10:40','11-21-2017 10:40')
- INSERT INTO BOR_Transaction VALUES(3039,'2017-11-21 10:20:00.000','2017-12-20 10:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3040,'2017-11-21 10:00:00.000','2017-12-27 10:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3041,'2017-01-03 09:20:00.000','2017-02-13 09:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3042,'2017-01-13 17:00:00.000','2017-02-20 17:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3043,'2017-01-23 16:40:00.000','2017-02-21 16:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3044,'2017-01-24 16:20:00.000','2017-05-16 16:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3045,'2017-02-02 16:00:00.000','2017-05-23 16:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3046,'2017-02-09 15:40:00.000','2017-05-25 15:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3047,'2017-03-06 15:20:00.000','2017-06-12 15:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3048,'2017-03-24 15:00:00.000','2017-06-16 15:00:00.000')
- INSERT INTO BOR_Transaction VALUES(3049,'2017-04-06 14:20:00.000','2017-06-19 14:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3050,'2017-05-11 13:40:00.000','2017-06-30 13:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3051,'2017-06-14 13:20:00.000','2017-07-18 13:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3052,'7-24-2017 13:00','7-28-2017 13:00')
- INSERT INTO BOR_Transaction VALUES(3053,'8-23-2017 12:40','9-8-2017 12:40')
- INSERT INTO BOR_Transaction VALUES(3054,'9-13-2017 12:20','9-19-2017 12:20')
- INSERT INTO BOR_Transaction VALUES(3055,'9-15-2017 12:00','9-25-2017 12:00')
- INSERT INTO BOR_Transaction VALUES(3056,'2017-09-25 11:40:00.000','2017-10-13 11:40:00.000')
- INSERT INTO BOR_Transaction VALUES(3057,'2017-09-25 11:20:00.000','2017-10-23 11:20:00.000')

- INSERT INTO BOR_Transaction VALUES(3058,'11-20-2017 10:40','11-21-2017 10:40')
- INSERT INTO BOR_Transaction VALUES(3059,'2017-11-21 10:20:00.000','2017-12-20 10:20:00.000')
- INSERT INTO BOR_Transaction VALUES(3060,'2017-11-21 10:00:00.000','2017-12-27 10:00:00.000')
-
- --Reservation
-
- INSERT INTO Reservation VALUES(4001,'5-11-2017 13:40','Cancelled')
- INSERT INTO Reservation VALUES(4002,'6-30-2017 13:40', 'Reserved')
- INSERT INTO Reservation VALUES(4003,'9-19-2017 12:20', 'Cancelled')
- INSERT INTO Reservation VALUES(4004,'11-21-2017 10:00', 'Cancelled')
- INSERT INTO Reservation VALUES(4005,'6-14-2017 13:20', 'Reserved')
- INSERT INTO Reservation VALUES(4006,'5-23-2017 16:00', 'Reserved')
- INSERT INTO Reservation VALUES(4007,'2-2-2017 16:00', 'Reserved')
- INSERT INTO Reservation VALUES(4008,'2-21-2017 16:40', 'Cancelled')
- INSERT INTO Reservation VALUES(4009,'1-3-2017 09:20', 'Reserved')
- INSERT INTO Reservation VALUES(4010,'3-6-2017 15:20', 'Cancelled')
- INSERT INTO Reservation VALUES(4011,'3-6-2017 15:20', 'Reserved')
- INSERT INTO Reservation VALUES(4012,'3-6-2017 15:20', 'Cancelled')
- INSERT INTO Reservation VALUES(4013,'9-15-2017 12:00', 'Reserved')
- INSERT INTO Reservation VALUES(4014,'1-23-2017 16:40', 'Reserved')
- INSERT INTO Reservation VALUES(4015,'5-25-2017 15:40', 'Reserved')
- INSERT INTO Reservation VALUES(4016,'10-23-2017 11:20', 'Cancelled')
- INSERT INTO Reservation VALUES(4017,'2-2-2017 16:00', 'Reserved')
- INSERT INTO Reservation VALUES(4018,'2-2-2017 16:00', 'Reserved')
- INSERT INTO Reservation VALUES(4019,'7-28-2017 13:00', 'Cancelled')
- INSERT INTO Reservation VALUES(4020,'7-28-2017 13:00', 'Reserved')
- INSERT INTO Reservation VALUES(4021,'1-12-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4022,'2-14-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4023,'3-18-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4024,'4-24-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4025,'5-15-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4026,'2-6-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4027,'5-7-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4028,'9-8-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4029,'4-9-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4030,'5-10-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4031,'6-12-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4032,'7-11-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4033,'1-13-2018 13:40','Canceled')
- INSERT INTO Reservation VALUES(4034,'2-27-2018 13:40','Reserved')
- INSERT INTO Reservation VALUES(4035,'3-31-2018 13:40','Canceled')

```
o   INSERT INTO Reservation VALUES(4036,'4-4-2018 13:40','Reserved')
o   INSERT INTO Reservation VALUES(4037,'5-5-2018 13:40','Canceled')
o   INSERT INTO Reservation VALUES(4038,'7-6-2018 13:40','Reserved')
o   INSERT INTO Reservation VALUES(4039,'2-7-2018 13:40','Canceled')
o   INSERT INTO Reservation VALUES(4040,'8-15-2018 13:40','Reserved')
o
o   --Chief editor
o   INSERT INTO Chief_Editor VALUES(4001,'Lucille   Banks')
o   INSERT INTO Chief_Editor VALUES(4002,'Sarah     Cain')
o   INSERT INTO Chief_Editor VALUES(4003,'Sherri    Mckinney')
o   INSERT INTO Chief_Editor VALUES(4004,'Mario     Santiago')
o   INSERT INTO Chief_Editor VALUES(4005,'Gertrude          Howell')
o   INSERT INTO Chief_Editor VALUES(4006,'Ross      Rogers')
o   INSERT INTO Chief_Editor VALUES(4007,'Grady     Drake')
o   INSERT INTO Chief_Editor VALUES(4008,'Angelo    Schneider')
o   INSERT INTO Chief_Editor VALUES(4009,'Debbie    Wells')
o   INSERT INTO Chief_Editor VALUES(4010,'Melody    West')
o   INSERT INTO Chief_Editor VALUES(4011,'Terrance          Massey')
o   INSERT INTO Chief_Editor VALUES(4012,'Misty     Lowe')
o   INSERT INTO Chief_Editor VALUES(4013,'Caleb     Mcguire')
o   INSERT INTO Chief_Editor VALUES(4014,'Kurt      Patton')
o   INSERT INTO Chief_Editor VALUES(4015,'Kari      Fleming')
o   INSERT INTO Chief_Editor VALUES(4016,'Teri      Barker')
o   INSERT INTO Chief_Editor VALUES(4017,'Herman            Castro')
o   INSERT INTO Chief_Editor VALUES(4018,'Kendra    Barrett')
o   INSERT INTO Chief_Editor VALUES(4019,'Martha    Malone')
o   INSERT INTO Chief_Editor VALUES(4020,'Marco     Fitzgerald')
o
o   --Author
o   INSERT INTO Author VALUES(5001,'Penelope    Harrell')
o   INSERT INTO Author VALUES(5002,'Ching Cheatham')
o   INSERT INTO Author VALUES(5003,'Alvera      Cambell')
o   INSERT INTO Author VALUES(5004,'Dannette    Foust')
o   INSERT INTO Author VALUES(5005,'Corrinne    Hayward')
o   INSERT INTO Author VALUES(5006,'Reynaldo    Whitworth')
o   INSERT INTO Author VALUES(5007,'Reuben      Sweeney')
o   INSERT INTO Author VALUES(5008,'Leonora     Roche')
o   INSERT INTO Author VALUES(5009,'Noella      Bearden')
o   INSERT INTO Author VALUES(5010,'Kyung       Ragan')
o   INSERT INTO Author VALUES(5011,'Elinor Chilton')
o   INSERT INTO Author VALUES(5012,'Tifany Cornish')
o   INSERT INTO Author VALUES(5013,'Dwana       Hollins')
o   INSERT INTO Author VALUES(5014,'Shirely     Dobson')
o   INSERT INTO Author VALUES(5015,'Cedrick     Covington')
```

```
o   INSERT INTO Author VALUES(5016,'Mertie Harder')
o   INSERT INTO Author VALUES(5017,'Lien    Vue')
o   INSERT INTO Author VALUES(5018,'Darleen         Schmitt')
o   INSERT INTO Author VALUES(5019,'Cammie          Kline')
o   INSERT INTO Author VALUES(5020,'Jadwiga         Belton')
o
o   --Document
o
o   INSERT INTO Document VALUES(6001,'abc','3-23-2009','2001')
o   INSERT INTO Document VALUES(6002,'def','7-7-2009','2010')
o   INSERT INTO Document VALUES(6003,'ghi','4-1-2010','2015')
o   INSERT INTO Document VALUES(6004,'jkl','6-30-2010','2002')
o   INSERT INTO Document VALUES(6005,'mno','9-28-2010','2001')
o   INSERT INTO Document VALUES(6006,'pqr','12-15-2010','2005')
o   INSERT INTO Document VALUES(6007,'stu','3-16-2011','2006')
o   INSERT INTO Document VALUES(6008,'vwx','3-22-2011','2017')
o   INSERT INTO Document VALUES(6009,'yzz','4-13-2011','2020')
o   INSERT INTO Document VALUES(6010,'ab1','5-19-2011','2020')
o   INSERT INTO Document VALUES(6011,'cd2','9-27-2011','2011')
o   INSERT INTO Document VALUES(6012,'ef3','2-22-2012','2014')
o   INSERT INTO Document VALUES(6013,'gh4','7-12-2012','2015')
o   INSERT INTO Document VALUES(6014,'ij5','3-18-2014','2016')
o   INSERT INTO Document VALUES(6015,'kl6','5-12-2014','2018')
o   INSERT INTO Document VALUES(6016,'mn7','11-3-2014','2020')
o   INSERT INTO Document VALUES(6017,'op8','3-5-2015','2009')
o   INSERT INTO Document VALUES(6018,'qr9','5-26-2015','2020')
o   INSERT INTO Document VALUES(6019,'st10','7-21-2015','2019')
o   INSERT INTO Document VALUES(6020,'st11','4-19-2017','2006')
o   INSERT INTO Document VALUES(6021,'abc','3-23-2009','2001')
o   INSERT INTO Document VALUES(6022,'def','7-7-2009','2010')
o   INSERT INTO Document VALUES(6023,'ghi','4-1-2010','2015')
o   INSERT INTO Document VALUES(6024,'jkl','6-30-2010','2002')
o   INSERT INTO Document VALUES(6025,'mno','9-28-2010','2001')
o   INSERT INTO Document VALUES(6026,'pqr','12-15-2010','2005')
o   INSERT INTO Document VALUES(6027,'stu','3-16-2011','2006')
o   INSERT INTO Document VALUES(6028,'vwx','3-22-2011','2017')
o   INSERT INTO Document VALUES(6029,'yzz','4-13-2011','2020')
o   INSERT INTO Document VALUES(6030,'ab1','5-19-2011','2020')
o   INSERT INTO Document VALUES(6031,'cd2','9-27-2011','2011')
o   INSERT INTO Document VALUES(6032,'ef3','2-22-2012','2014')
o   INSERT INTO Document VALUES(6033,'gh4','7-12-2012','2015')
o   INSERT INTO Document VALUES(6034,'ij5','3-18-2014','2016')
o   INSERT INTO Document VALUES(6035,'kl6','5-12-2014','2018')
o   INSERT INTO Document VALUES(6036,'mn7','11-3-2014','2020')
```

- INSERT INTO Document VALUES(6037,'op8','3-5-2015','2009')
- INSERT INTO Document VALUES(6038,'qr9','5-26-2015','2020')
- INSERT INTO Document VALUES(6039,'st10','7-21-2015','2019')
- INSERT INTO Document VALUES(6040,'st11','4-19-2017','2006')
- INSERT INTO Document VALUES(6041,'abc','3-23-2009','2001')
- INSERT INTO Document VALUES(6042,'def','7-7-2009','2010')
- INSERT INTO Document VALUES(6043,'ghi','4-1-2010','2015')
- INSERT INTO Document VALUES(6044,'jkl','6-30-2010','2002')
- INSERT INTO Document VALUES(6045,'mno','9-28-2010','2001')
- INSERT INTO Document VALUES(6046,'pqr','12-15-2010','2005')
- INSERT INTO Document VALUES(6047,'stu','3-16-2011','2006')
- INSERT INTO Document VALUES(6048,'vwx','3-22-2011','2017')
- INSERT INTO Document VALUES(6049,'yzz','4-13-2011','2020')
- INSERT INTO Document VALUES(6050,'ab1','5-19-2011','2020')
- INSERT INTO Document VALUES(6051,'cd2','9-27-2011','2011')
- INSERT INTO Document VALUES(6052,'ef3','2-22-2012','2014')
- INSERT INTO Document VALUES(6053,'gh4','7-12-2012','2015')
- INSERT INTO Document VALUES(6054,'ij5','3-18-2014','2016')
- INSERT INTO Document VALUES(6055,'kl6','5-12-2014','2018')
- INSERT INTO Document VALUES(6056,'mn7','11-3-2014','2020')
- INSERT INTO Document VALUES(6057,'op8','3-5-2015','2009')
- INSERT INTO Document VALUES(6058,'qr9','5-26-2015','2020')
- INSERT INTO Document VALUES(6059,'st10','7-21-2015','2019')
- INSERT INTO Document VALUES(6060,'st11','4-19-2017','2006')
-
-
- --book
- INSERT INTO book VALUES(6010,'7197364657922')
- INSERT INTO book VALUES(6009,'9858110330262')
- INSERT INTO book VALUES(6008,'0175037910599')
- INSERT INTO book VALUES(6001,'4991257779220')
- INSERT INTO book VALUES(6020,'9982792269430')
- INSERT INTO book VALUES(6003,'8553472733173')
- INSERT INTO book VALUES(6005,'4684453129678')
- INSERT INTO book VALUES(6013,'3832743195028')
- INSERT INTO book VALUES(6006,'7647302241110')
- INSERT INTO book VALUES(6019,'1841808291880')
- INSERT INTO book VALUES(6007,'7577449673602')
- INSERT INTO book VALUES(6015,'7176512535629')
- INSERT INTO book VALUES(6004,'9971720137252')
- INSERT INTO book VALUES(6021,'8989265265841')
- INSERT INTO book VALUES(6018,'1451544384755')
- INSERT INTO book VALUES(6002,'1386901361258')
- INSERT INTO book VALUES(6017,'8109173267105')

- INSERT INTO book VALUES(6011,'3693558915201')
- INSERT INTO book VALUES(6012,'0539787786878')
- INSERT INTO book VALUES(6014,'5469761650868')
- INSERT INTO book VALUES(6016,'2178065428658')
-
- --writes
- INSERT INTO writes VALUES(6001,5020)
- INSERT INTO writes VALUES(6002,5019)
- INSERT INTO writes VALUES(6003,5018)
- INSERT INTO writes VALUES(6004,5017)
- INSERT INTO writes VALUES(6005,5016)
- INSERT INTO writes VALUES(6006,5015)
- INSERT INTO writes VALUES(6007,5014)
- INSERT INTO writes VALUES(6008,5013)
- INSERT INTO writes VALUES(6009,5012)
- INSERT INTO writes VALUES(6010,5011)
- INSERT INTO writes VALUES(6011,5010)
- INSERT INTO writes VALUES(6012,5009)
- INSERT INTO writes VALUES(6013,5008)
- INSERT INTO writes VALUES(6014,5007)
- INSERT INTO writes VALUES(6015,5006)
- INSERT INTO writes VALUES(6016,5005)
- INSERT INTO writes VALUES(6017,5004)
- INSERT INTO writes VALUES(6018,5003)
- INSERT INTO writes VALUES(6019,5002)
- INSERT INTO writes VALUES(6020,5001)
-
- --proceedings
-
- INSERT INTO Proceedings VALUES(6021,'1-18-2016','Sucker Flat, Georgia','Eileen Morrison')
- INSERT INTO Proceedings VALUES(6022,'2-1-2016','Wawawai, Kansas','Robert Greene')
- INSERT INTO Proceedings VALUES(6023,'4-7-2016','Totopitk, Rhode Island','Sharon Wood')
- INSERT INTO Proceedings VALUES(6024,'6-24-2016','Buckskin Joe, Texas','Alexander Salazar')
- INSERT INTO Proceedings VALUES(6025,'7-5-2016','Fairplay, South Dakota','Albert Little')
- INSERT INTO Proceedings VALUES(6026,'9-2-2016','Hygiene, New Mexico','Lorraine Hines')
- INSERT INTO Proceedings VALUES(6027,'9-9-2016','Severance, Arkansas','Jesus Cunningham')

- INSERT INTO Proceedings VALUES(6028,'11-16-2016','Thoreau, Georgia','Kristopher Davis')
- INSERT INTO Proceedings VALUES(6029,'11-30-2016','Moons, Hawaii','Frederick Craig')
- INSERT INTO Proceedings VALUES(6030,'12-5-2016','Love Lady, Alabama','Minnie Thompson')
- INSERT INTO Proceedings VALUES(6031,'12-9-2016','Tugwell, Kansas','Steve Curtis')
- INSERT INTO Proceedings VALUES(6032,'2-21-2017','Tickfaw, Rhode Island','Timmy Lawson')
- INSERT INTO Proceedings VALUES(6033,'3-22-2017','Kaskaskia, Maine','Sean Miles')
- INSERT INTO Proceedings VALUES(6034,'3-23-2017','Duckers, Iowa','Darlene Hampton')
- INSERT INTO Proceedings VALUES(6035,'5-4-2017','Pennsylvania, Maine','Nicole Kim')
- INSERT INTO Proceedings VALUES(6036,'5-8-2017','Economy, Maine','Jon Fitzgerald')
- INSERT INTO Proceedings VALUES(6037,'6-16-2017','Graphic, Kansas','Noel King')
- INSERT INTO Proceedings VALUES(6038,'7-6-2017','Bondurant, New Mexico','Ernest Jordan')
- INSERT INTO Proceedings VALUES(6039,'11-22-2017','Ransom, Arkansas','Fredrick Webster')
- INSERT INTO Proceedings VALUES(6040,'12-4-2017','Morning Star, North Dakota','Ross Phillips')
-
- --journal volume
-
- INSERT INTO Journal_Volume VALUES(6041,7001,4001)
- INSERT INTO Journal_Volume VALUES(6042,7002,4005)
- INSERT INTO Journal_Volume VALUES(6043,7003,4006)
- INSERT INTO Journal_Volume VALUES(6044,7004,4009)
- INSERT INTO Journal_Volume VALUES(6045,7005,4007)
- INSERT INTO Journal_Volume VALUES(6046,7006,4001)
- INSERT INTO Journal_Volume VALUES(6047,7007,4017)
- INSERT INTO Journal_Volume VALUES(6048,7008,4010)
- INSERT INTO Journal_Volume VALUES(6049,7009,4016)
- INSERT INTO Journal_Volume VALUES(6050,7010,4015)
- INSERT INTO Journal_Volume VALUES(6051,7011,4019)
- INSERT INTO Journal_Volume VALUES(6052,7012,4020)
- INSERT INTO Journal_Volume VALUES(6053,7013,4020)
- INSERT INTO Journal_Volume VALUES(6054,7014,4017)
- INSERT INTO Journal_Volume VALUES(6055,7015,4011)

```
INSERT INTO Journal_Volume VALUES(6056,7016,4002)
INSERT INTO Journal_Volume VALUES(6057,7017,4002)
INSERT INTO Journal_Volume VALUES(6058,7018,4009)
INSERT INTO Journal_Volume VALUES(6059,7019,4003)
INSERT INTO Journal_Volume VALUES(6060,7020,4004)

--Journal issue
INSERT INTO Journal_Issue VALUES(6041,8001,'0-46')
INSERT INTO Journal_Issue VALUES(6042,8002,'24-38')
INSERT INTO Journal_Issue VALUES(6043,8003,'56-78')
INSERT INTO Journal_Issue VALUES(6044,8004,'2-19')
INSERT INTO Journal_Issue VALUES(6045,8005,'2-78')
INSERT INTO Journal_Issue VALUES(6046,8006,'67-99')
INSERT INTO Journal_Issue VALUES(6047,8007,'26-44')
INSERT INTO Journal_Issue VALUES(6048,8008,'13-89')
INSERT INTO Journal_Issue VALUES(6049,8009,'78-99')
INSERT INTO Journal_Issue VALUES(6050,8010,'15-78')
INSERT INTO Journal_Issue VALUES(6051,8011,'45-56')
INSERT INTO Journal_Issue VALUES(6052,8012,'57-68')
INSERT INTO Journal_Issue VALUES(6053,8013,'1-10')
INSERT INTO Journal_Issue VALUES(6054,8014,'8-19')
INSERT INTO Journal_Issue VALUES(6055,8015,'7-20')
INSERT INTO Journal_Issue VALUES(6056,8016,'3-14')
INSERT INTO Journal_Issue VALUES(6057,8017,'14-47')
INSERT INTO Journal_Issue VALUES(6058,8018,'9-12')
INSERT INTO Journal_Issue VALUES(6059,8019,'19-29')
INSERT INTO Journal_Issue VALUES(6060,8020,'55-70')

--Guest_Editor

INSERT INTO Guest_Editor VALUES(6041,8001,'Leona	Briggs')
INSERT INTO Guest_Editor VALUES(6042,8002,'Jesse	Perkins')
INSERT INTO Guest_Editor VALUES(6043,8003,'Cary	Warner')
INSERT INTO Guest_Editor VALUES(6044,8004,'Jerome	Hart')
INSERT INTO Guest_Editor VALUES(6045,8005,'Myrtle	Bush')
INSERT INTO Guest_Editor VALUES(6046,8006,'Julie	Rodriguez')
INSERT INTO Guest_Editor VALUES(6047,8007,'Maria	Underwood')
INSERT INTO Guest_Editor VALUES(6048,8008,'Arlene	Carlson')
INSERT INTO Guest_Editor VALUES(6049,8009,'Nick	Reese')
INSERT INTO Guest_Editor VALUES(6050,8010,'Rosie	Carter')
INSERT INTO Guest_Editor VALUES(6051,8011,'Esther	Delgado')
INSERT INTO Guest_Editor VALUES(6052,8012,'Jeanne	Hopkins')
INSERT INTO Guest_Editor VALUES(6053,8013,'Luke	Ramsey')
INSERT INTO Guest_Editor VALUES(6054,8014,'Carrie	Dean')
INSERT INTO Guest_Editor VALUES(6055,8015,'Dale      Wade')
INSERT INTO Guest_Editor VALUES(6056,8016,'Jared     Douglas')
INSERT INTO Guest_Editor VALUES(6057,8017,'Christie  Peterson')
INSERT INTO Guest_Editor VALUES(6058,8018,'Matt      Romero')
INSERT INTO Guest_Editor VALUES(6059,8019,'Ernestine         Lambert')
INSERT INTO Guest_Editor VALUES(6060,8020,'Juan      Olson')

--copy
INSERT INTO Copy VALUES (1,6001,9001,'012D03')
INSERT INTO Copy VALUES (1,6002,9002,'461Q78')
INSERT INTO Copy VALUES (2,6003,9003,'040U83')
INSERT INTO Copy VALUES (3,6004,9004,'003S06')
INSERT INTO Copy VALUES (4,6005,9005,'754J64')
INSERT INTO Copy VALUES (4,6006,9006,'002M68')
INSERT INTO Copy VALUES (5,6007,9007,'365J97')
INSERT INTO Copy VALUES (6,6008,9008,'193F78')
INSERT INTO Copy VALUES (8,6009,9009,'321K89')
INSERT INTO Copy VALUES (9,6010,9010,'465L09')
INSERT INTO Copy VALUES (14,6011,9011,'100T73')
INSERT INTO Copy VALUES (15,6012,9012,'183O23')
INSERT INTO Copy VALUES (17,6013,9013,'456Y38')
INSERT INTO Copy VALUES (17,6014,9014,'009X75')
INSERT INTO Copy VALUES (19,6015,9015,'999Z71')
INSERT INTO Copy VALUES (19,6016,9016,'128R69')
INSERT INTO Copy VALUES (20,6017,9017,'227W22')
INSERT INTO Copy VALUES (1,6018,9018,'012D03')
INSERT INTO Copy VALUES (1,6019,9019,'461Q78')
INSERT INTO Copy VALUES (2,6020,9020,'040U83')
INSERT INTO Copy VALUES (3,6021,9021,'003S06')
INSERT INTO Copy VALUES (4,6022,9022,'754J64')
INSERT INTO Copy VALUES (4,6023,9023,'002M68')
INSERT INTO Copy VALUES (5,6024,9024,'365J97')
INSERT INTO Copy VALUES (6,6025,9025,'193F78')
INSERT INTO Copy VALUES (8,6026,9026,'321K89')
INSERT INTO Copy VALUES (9,6027,9027,'465L09')
INSERT INTO Copy VALUES (14,6028,9028,'100T73')
INSERT INTO Copy VALUES (15,6029,9029,'183O23')
INSERT INTO Copy VALUES (17,6030,9030,'456Y38')
INSERT INTO Copy VALUES (17,6031,9031,'009X75')
INSERT INTO Copy VALUES (19,6032,9032,'999Z71')
INSERT INTO Copy VALUES (19,6033,9033,'128R69')
INSERT INTO Copy VALUES (20,6034,9034,'227W22')
INSERT INTO Copy VALUES (1,6035,9035,'012D03')
INSERT INTO Copy VALUES (1,6036,9036,'461Q78')
```

- INSERT INTO Copy VALUES (2,6037,9037,'040U83')
- INSERT INTO Copy VALUES (3,6038,9038,'003S06')
- INSERT INTO Copy VALUES (4,6039,9039,'754J64')
- INSERT INTO Copy VALUES (4,6040,9040,'002M68')
- INSERT INTO Copy VALUES (5,6041,9041,'365J97')
- INSERT INTO Copy VALUES (6,6042,9042,'193F78')
- INSERT INTO Copy VALUES (8,6043,9043,'321K89')
- INSERT INTO Copy VALUES (9,6044,9044,'465L09')
- INSERT INTO Copy VALUES (14,6045,9045,'100T73')
- INSERT INTO Copy VALUES (15,6046,9046,'183O23')
- INSERT INTO Copy VALUES (17,6047,9047,'456Y38')
- INSERT INTO Copy VALUES (17,6048,9048,'009X75')
- INSERT INTO Copy VALUES (19,6049,9049,'999Z71')
- INSERT INTO Copy VALUES (19,6050,9050,'128R69')
- INSERT INTO Copy VALUES (20,6051,9051,'227W22')
- INSERT INTO Copy VALUES (6,6052,9052,'365J97')
- INSERT INTO Copy VALUES (8,6053,9053,'193F78')
- INSERT INTO Copy VALUES (9,6054,9054,'321K89')
- INSERT INTO Copy VALUES (14,6055,9055,'465L09')
- INSERT INTO Copy VALUES (15,6056,9056,'100T73')
- INSERT INTO Copy VALUES (17,6057,9057,'183O23')
- INSERT INTO Copy VALUES (17,6058,9058,'456Y38')
- INSERT INTO Copy VALUES (19,6059,9059,'009X75')
- INSERT INTO Copy VALUES (19,6060,9060,'999Z71')
- INSERT INTO Copy VALUES (1,6032,9001,'012D03')
- INSERT INTO Copy VALUES (2,6013,9002,'461Q78')
- INSERT INTO Copy VALUES (4,6054,9003,'040U83')
- INSERT INTO Copy VALUES (6,6013,9004,'003S06')
- INSERT INTO Copy VALUES (8,6024,9005,'012D03')
- INSERT INTO Copy VALUES (4,6053,9006,'012D03')
- INSERT INTO Copy VALUES (3,6013,9007,'461Q78')
- INSERT INTO Copy VALUES (6,6011,9008,'040U83')
- INSERT INTO Copy VALUES (3,6001,9009,'003S06')
- INSERT INTO Copy VALUES (11,6003,9010,'012D03')
- INSERT INTO Copy VALUES (14,6004,9011,'012D03')
- INSERT INTO Copy VALUES (20,6005,9012,'461Q78')
- INSERT INTO Copy VALUES (12,6006,9013,'040U83')
- INSERT INTO Copy VALUES (15,6007,9014,'003S06')
- INSERT INTO Copy VALUES (17,6006,9015,'012D03')
- INSERT INTO Copy VALUES (7,6003,9016,'012D03')
- INSERT INTO Copy VALUES (7,6004,9017,'461Q78')
- INSERT INTO Copy VALUES (8,6002,9018,'040U83')
- INSERT INTO Copy VALUES (4,6013,9019,'003S06')
- INSERT INTO Copy VALUES (3,6025,9020,'012D03')

- INSERT INTO Copy VALUES (3,6043,9021,'012D03')
- INSERT INTO Copy VALUES (5,6052,9022,'461Q78')
- INSERT INTO Copy VALUES (8,6024,9023,'040U83')
- INSERT INTO Copy VALUES (5,6013,9024,'003S06')
- INSERT INTO Copy VALUES (4,6025,9025,'012D03')
- INSERT INTO Copy VALUES (2,6035,9026,'012D03')
- INSERT INTO Copy VALUES (4,6023,9027,'461Q78')
- INSERT INTO Copy VALUES (6,6013,9028,'040U83')
- INSERT INTO Copy VALUES (7,6035,9029,'003S06')
- INSERT INTO Copy VALUES (3,6057,9030,'012D03')
- INSERT INTO Copy VALUES (6,6024,9031,'012D03')
- INSERT INTO Copy VALUES (8,6013,9032,'461Q78')
- INSERT INTO Copy VALUES (9,6024,9033,'040U83')
- INSERT INTO Copy VALUES (9,6046,9034,'003S06')
- INSERT INTO Copy VALUES (8,6057,9035,'012D03')
- INSERT INTO Copy VALUES (7,6006,9036,'012D03')
- INSERT INTO Copy VALUES (14,6042,9037,'461Q78')
- INSERT INTO Copy VALUES (3,6031,9038,'040U83')
- INSERT INTO Copy VALUES (19,6024,9039,'003S06')
- INSERT INTO Copy VALUES (18,6046,9040,'012D03')
-
-
- --Reserves
-
- INSERT INTO Reserves VALUES (4001,6021,3,9021,1001)
- INSERT INTO Reserves VALUES (4002,6022,4,9022,1008)
- INSERT INTO Reserves VALUES (4003,6023,4,9023,1009)
- INSERT INTO Reserves VALUES (4004,6024,5,9024,1007)
- INSERT INTO Reserves VALUES (4005,6025,6,9025,1004)
- INSERT INTO Reserves VALUES (4006,6026,8,9026,1003)
- INSERT INTO Reserves VALUES (4007,6027,9,9027,1001)
- INSERT INTO Reserves VALUES (4008,6028,14,9028,1005)
- INSERT INTO Reserves VALUES (4009,6029,15,9029,1010)
- INSERT INTO Reserves VALUES (4010,6030,17,9030,1011)
- INSERT INTO Reserves VALUES (4011,6031,17,9031,1019)
- INSERT INTO Reserves VALUES (4012,6032,19,9032,1020)
- INSERT INTO Reserves VALUES (4013,6033,19,9033,1020)
- INSERT INTO Reserves VALUES (4014,6034,20,9034,1020)
- INSERT INTO Reserves VALUES (4015,6035,1,9035,1017)
- INSERT INTO Reserves VALUES (4016,6036,1,9036,1019)
- INSERT INTO Reserves VALUES (4017,6037,2,9037,1016)
- INSERT INTO Reserves VALUES (4018,6038,3,9038,1015)
- INSERT INTO Reserves VALUES (4019,6039,4,9039,1015)
- INSERT INTO Reserves VALUES (4020,6040,4,9040,1020)

- INSERT INTO Reserves VALUES (4021,6041,5,9041,1001)
- INSERT INTO Reserves VALUES (4022,6042,6,9042,1002)
- INSERT INTO Reserves VALUES (4023,6043,8,9043,1003)
- INSERT INTO Reserves VALUES (4024,6044,9,9044,1004)
- INSERT INTO Reserves VALUES (4025,6045,14,9045,1005)
- INSERT INTO Reserves VALUES (4026,6046,15,9046,1006)
- INSERT INTO Reserves VALUES (4027,6047,17,9047,1007)
- INSERT INTO Reserves VALUES (4028,6048,17,9048,1008)
- INSERT INTO Reserves VALUES (4029,6049,19,9049,1009)
- INSERT INTO Reserves VALUES (4030,6050,19,9050,1010)
- INSERT INTO Reserves VALUES (4031,6051,20,9051,1011)
- INSERT INTO Reserves VALUES (4032,6052,6,9052,1012)
- INSERT INTO Reserves VALUES (4033,6053,8,9053,1013)
- INSERT INTO Reserves VALUES (4034,6054,9,9054,1014)
- INSERT INTO Reserves VALUES (4035,6055,14,9055,1015)
- INSERT INTO Reserves VALUES (4036,6056,15,9056,1016)
- INSERT INTO Reserves VALUES (4037,6057,17,9057,1017)
- INSERT INTO Reserves VALUES (4038,6058,17,9058,1018)
- INSERT INTO Reserves VALUES (4039,6059,19,9059,1019)
- INSERT INTO Reserves VALUES (4040,6060,19,9060,1020)
-
- --borrows
- INSERT INTO Borrows VALUES (3001,6001,1,9001,1001)
- INSERT INTO Borrows VALUES (3002,6002,1,9002,1008)
- INSERT INTO Borrows VALUES (3003,6003,2,9003,1009)
- INSERT INTO Borrows VALUES (3004,6004,3,9004,1007)
- INSERT INTO Borrows VALUES (3005,6005,4,9005,1004)
- INSERT INTO Borrows VALUES (3006,6006,4,9006,1003)
- INSERT INTO Borrows VALUES (3007,6007,5,9007,1001)
- INSERT INTO Borrows VALUES (3008,6008,6,9008,1005)
- INSERT INTO Borrows VALUES (3009,6009,8,9009,1010)
- INSERT INTO Borrows VALUES (3010,6010,9,9010,1011)
- INSERT INTO Borrows VALUES (3011,6011,14,9011,1019)
- INSERT INTO Borrows VALUES (3012,6012,15,9012,1020)
- INSERT INTO Borrows VALUES (3013,6013,17,9013,1020)
- INSERT INTO Borrows VALUES (3014,6014,17,9014,1020)
- INSERT INTO Borrows VALUES (3015,6015,19,9015,1017)
- INSERT INTO Borrows VALUES (3016,6016,19,9016,1019)
- INSERT INTO Borrows VALUES (3017,6017,20,9017,1016)
- INSERT INTO Borrows VALUES (3018,6018,1,9018,1015)
- INSERT INTO Borrows VALUES (3019,6019,1,9019,1015)
- INSERT INTO Borrows VALUES (3020,6020,2,9020,1020)
- INSERT INTO Borrows VALUES (3021,6032,1,9001,1005)
- INSERT INTO Borrows VALUES (3022,6013,2,9002,1014)

    - INSERT INTO Borrows VALUES (3023,6054,4,9003,1002)
    - INSERT INTO Borrows VALUES (3024,6013,6,9004,1001)
    - INSERT INTO Borrows VALUES (3025,6024,8,9005,1004)
    - INSERT INTO Borrows VALUES (3026,6053,4,9006,1005)
    - INSERT INTO Borrows VALUES (3027,6013,3,9007,1007)
    - INSERT INTO Borrows VALUES (3028,6011,6,9008,1002)
    - INSERT INTO Borrows VALUES (3029,6001,3,9009,1009)
    - INSERT INTO Borrows VALUES (3030,6003,11,9010,1015)
    - INSERT INTO Borrows VALUES (3031,6004,14,9011,1011)
    - INSERT INTO Borrows VALUES (3032,6005,20,9012,1012)
    - INSERT INTO Borrows VALUES (3033,6006,12,9013,1017)
    - INSERT INTO Borrows VALUES (3034,6007,15,9014,1018)
    - INSERT INTO Borrows VALUES (3035,6006,17,9015,1020)
    - INSERT INTO Borrows VALUES (3036,6003,7,9016,1019)
    - INSERT INTO Borrows VALUES (3037,6004,7,9017,1006)
    - INSERT INTO Borrows VALUES (3038,6002,8,9018,1012)
    - INSERT INTO Borrows VALUES (3039,6013,4,9019,1006)
    - INSERT INTO Borrows VALUES (3040,6025,3,9020,1008)
    - INSERT INTO Borrows VALUES (3041,6043,3,9021,1009)
    - INSERT INTO Borrows VALUES (3042,6052,5,9022,1005)
    - INSERT INTO Borrows VALUES (3043,6024,8,9023,1014)
    - INSERT INTO Borrows VALUES (3044,6013,5,9024,1002)
    - INSERT INTO Borrows VALUES (3045,6025,4,9025,1001)
    - INSERT INTO Borrows VALUES (3046,6035,2,9026,1004)
    - INSERT INTO Borrows VALUES (3047,6023,4,9027,1005)
    - INSERT INTO Borrows VALUES (3048,6013,6,9028,1007)
    - INSERT INTO Borrows VALUES (3049,6035,7,9029,1002)
    - INSERT INTO Borrows VALUES (3050,6057,3,9030,1009)
    - INSERT INTO Borrows VALUES (3051,6024,6,9031,1015)
    - INSERT INTO Borrows VALUES (3052,6013,8,9032,1011)
    - INSERT INTO Borrows VALUES (3053,6024,9,9033,1012)
    - INSERT INTO Borrows VALUES (3054,6046,9,9034,1017)
    - INSERT INTO Borrows VALUES (3055,6057,8,9035,1018)
    - INSERT INTO Borrows VALUES (3056,6006,7,9036,1020)
    - INSERT INTO Borrows VALUES (3057,6042,14,9037,1019)
    - INSERT INTO Borrows VALUES (3058,6031,3,9038,1006)
    - INSERT INTO Borrows VALUES (3059,6024,19,9039,1012)
    - INSERT INTO Borrows VALUES (3060,6046,18,9040,1006)
- **spAddDocument:**
    - CREATE PROCEDURE spAddDocument(@docTitle nvarchar(50), @pdate date, @pubid INT, @libid int, @copyno int, @position nvarchar(10), @status INT out)
    - AS
    - DECLARE @count1 INT   -- to check if publisher exists
    - DECLARE @count2 INT   -- to check if libid exists

```
DECLARE @docId INT
BEGIN
        SET @docId=0
        select @count1=count(1) from Publisher where Publisher_id=@pubid
        if(@count1>0)
        BEGIN
                SELECT @count2=count(1) from Branch where Libid=@libid
                if(@count2>0)
                BEGIN
                        SELECT @docId=MAX(DocID) from Document
                        IF @docId is null
                                SET @docId = 6000
                        SET @docId+=1
                        INSERT INTO Document Values(@docId,@docTitle,@pdate,@pubid)
                        INSERT INTO Copy Values(@libid,@docId,@copyno,@position)
                        SET @status=1
                END
                else
                        SET @status=-1
        END
        else
                SET @status=0
END
```

- **spAddReader:**

```
CREATE PROCEDURE spAddReader (@readerName nvarchar(50), @type nvarchar(10), @phoneno nvarchar(12),@address nvarchar(100))
AS
DECLARE @readerId as INT
BEGIN
        SELECT @readerId=MAX(Reader_id) from Reader
        if @readerId is null
                SET @readerId = 1000
        SET @readerId += 1
        INSERT INTO Reader VALUES (@readerId, @type, @readerName, @phoneno, @address)
END
```

- **spCheckDocStatus:**

```
Create PROCEDURE spCheckDocStatus(@docId int, @libId int, @copyno int, @status int out)
AS
DECLARE @count1 INT;  --If document exists
DECLARE @count2 INT;  --If document is borrowed
```

- DECLARE @count3 INT; --If document is reserved
- BEGIN
- UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime<SYSDATETIME()
- if(DATEDIFF(Second, CAST(CURRENT_TIMESTAMP as time) , CAST('18:00:00' as time)) < 0) -- TO CHECK CURRENT TIME
- UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime=SYSDATETIME()
- SELECT @count1=count(1) from Document where DocId=@docId
- SELECT @count2=count(1) from Borrows B inner join BOR_Transaction BT on B.BorNumber=BT.BorNumber where DocID=@docId and libID=@libId and copyno=@copyno and (BorDateTime is not null and RetDateTime is null)
- SELECT @count3=count(1) from Reserves R inner join Reservation RV on R.ResNumber=RV.ResNumber where DocID=@docId and libID=@libId and copyno=@copyno and Reservation_Status='Reserved'
- if(@count1 > 0 and @count2 = 0 and @count3 = 0)
- SET @status = 1
- else if (@count1 = 0)
- SET @status = 0
- else if (@count2 > 0)
- SET @status = -1
- else if (@count3 > 0)
- SET @status = -2
- END

- **spCheckOut:**
  - CREATE PROCEDURE spCheckout(@readerId INT,
  - @docId INT,
  - @copyNo INT,
  - @libId INT,
  - @result INT out)
  - AS
  - DECLARE @count1 INT --To check if book exists
  - DECLARE @count2 INT --To check if someone has borrowed it
  - DECLARE @count3 INT --To check if someone else has reserved it
  - DECLARE @BORNumber INT
  - BEGIN
  - UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime<SYSDATETIME()
  - if(DATEDIFF(Second, CAST(CURRENT_TIMESTAMP as time) , CAST('18:00:00' as time)) < 0) -- TO CHECK CURRENT TIME
  - UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime=SYSDATETIME()
  - SELECT @count1=count(1) from Copy where DocID=@docId and Copyno=@copyNo and LibId=@libId

```
	if @count1>0
	begin
		SELECT @count2=count(1) from Borrows B inner join BOR_Transaction BT on B.BorNumber=BT.BorNumber WHERE DocID=@docId and Copyno=@copyNo and LibId=@libId and RetDateTime is null
		SELECT @count3=count(1) from  Reserves R inner join Reservation RV on R.ResNumber=Rv.ResNumber WHERE DocID=@docId and LibId=@libId and Copyno=@copyNo and ReaderId<>@readerId and Reservation_Status = 'Reserved'
		if @count2=0 and @count3=0
		begin
			SELECT @BORNumber=MAX(BORNumber) from Borrows
			IF @BORNumber is null
				SET @BORNumber=3000
			SET @BORNumber += 1
			INSERT INTO BOR_Transaction VALUES (@BORNumber,SYSDATETIME(),null)
			INSERT INTO Borrows VALUES (@BORNumber,@docId,@libId,@copyNo,@readerId)
			Update Reservation SET Reservation_Status = 'Cancelled' Where ResNumber in (SELECT ResNumber from Reserves where ReaderId=@readerId and DocID=@docId and LibId=@libId and Copyno=@copyNo)
			SET @result = @BORNumber

		end
		ELSE
		BEGIN
			SET @result = 0
		END
	end
	else
	begin
		SET @result = -1
	end
END
```

- **spGetReservedDocs:**

```
CREATE PROCEDURE spGetReservedDocs(@readerId as Int)
As
Begin
	UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime<SYSDATETIME()
```

- if(DATEDIFF(Second, CAST(CURRENT_TIMESTAMP as time) , CAST('18:00:00' as time)) < 0)   -- TO CHECK CURRENT TIME
- UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime=SYSDATETIME()
- select R.DocID, Title, Reservation_Status AS STATUS from Reserves R Inner Join Document D on R.DocID = D.DocId Left outer join Reservation RV on R.ResNumber=Rv.ResNumber where ReaderId = @readerId
- End

* **spReserve:**
  - CREATE PROCEDURE [dbo].[spReserve](@readerId INT,
  - @docId INT,
  - @copyNo INT,
  - @libId INT,
  - @resDate DATE,
  - @result INT out)
  - AS
  - DECLARE @count1 INT --To check if book exists
  - DECLARE @count2 INT --To check if someone has borrowed it
  - DECLARE @count3 INT --To check if someone else has reserved it
  - DECLARE @count4 INT --To check if user has already reserved 10 documents
  - DECLARE @ResNumber INT
  - BEGIN
  - UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime<SYSDATETIME()
  - if(DATEDIFF(Second, CAST(CURRENT_TIMESTAMP as time) , CAST('18:00:00' as time)) < 0)   -- TO CHECK CURRENT TIME
  - UPDATE Reservation SET Reservation_Status = 'Cancelled' where ResDateTime=SYSDATETIME()
  - SELECT @count1=count(1) from Copy where DocID=@docId and Copyno=@copyNo and LibId=@libId
  - if @count1>0
  - begin
  - SELECT @count2=count(1) from Borrows B inner join BOR_Transaction BT on B.BorNumber=BT.BorNumber WHERE DocID=@docId and Copyno=@copyNo and LibId=@libId and RetDateTime is null
  - SELECT @count3=count(1) from Reserves R inner join Reservation RV on R.ResNumber=RV.ResNumber where DocID=@docId and LibId=@libId and Copyno=@copyNo and Reservation_Status = 'Reserved'

  - if @count2=0 and @count3=0
  - begin
  - SELECT @count4=count(1) from Reserves R inner join Reservation RV on R.ResNumber=RV.ResNumber where ReaderId=@readerId and Reservation_Status = 'Reserved'

```
            if @count4<10
            begin
                SELECT @ResNumber=MAX(ResNumber) from Reservation
                IF @ResNumber is Null
                    SET @ResNumber=4000
                SET @ResNumber += 1
                INSERT INTO Reservation VALUES (@ResNumber,@resDate,'Reserved')
                INSERT INTO Reserves VALUES (@ResNumber,@docId,@libId,@copyNo,@readerId)
                SET @result = @ResNumber

            end
            else
            begin
                set @result = 0
            end
        end
        ELSE
        BEGIN
            SET @result = -1
        END
    end
    else
    begin
        SET @result = -2
    end
END
```

- **spReturn:**

```
CREATE PROCEDURE spReturn(@readerId INT,
@docId INT,
@copyNo INT,
@libId INT,
@result INT out)
AS
DECLARE @count1 INT
DECLARE @temp INT
BEGIN
    SELECT @count1=count(1) from Borrows B INNER JOIN BOR_Transaction BT on B.BorNumber=BT.BorNumber WHERE DocID=@docId and Copyno=@copyNo and LibId=@libId and RetDateTime is null and BorDateTime is not null
    IF @count1=1
```

```
        BEGIN
                Update BOR_Transaction set RetDateTime=SYSDATETIME() where BorNumber in (SELECT B.BorNumber from Borrows B INNER JOIN BOR_Transaction BT on B.BorNumber=BT.BorNumber WHERE DocID=@docId and Copyno=@copyNo and LibId=@libId and RetDateTime is null and BorDateTime is not null)
                IF @@ROWCOUNT = 1
                        SET @result = 1                                     -- Updated successfully
                ELSE
                        SET @result = 0                                     -- Error
        END
        ELSE
                SET @result = -1                                            -- No record found of borrowing
END
```

- **AdminController:**

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using CityLibrary.Models;
using System.Data;
using System.Text.RegularExpressions;

namespace CityLibrary.Controllers
{
    public class AdminController : Controller
    {
        private CityLibraryAdminDAL dal = new CityLibraryAdminDAL();
        [HttpGet]
        public ActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Index(string adminId, string adminPass)
        {
            if (!String.IsNullOrEmpty(adminId) && !String.IsNullOrEmpty(adminPass))
            {
                string id = ConfigurationManager.AppSettings["adminId"].ToString();
                string pass = ConfigurationManager.AppSettings["adminPass"].ToString();
                if (adminId.Equals(id) && adminPass.Equals(pass))
                {
                    HttpContext.Session["adminId"] = adminId;
                    HttpContext.Session["readerName"] = null;
                    HttpContext.Session["readerId"] = null;
                    return RedirectToAction("Main");
                }
                else
                {
                    return View();
                }
            }
            else
            {
                return View();
            }
        }
        [HttpGet]
        public ActionResult Main()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index","Home");
            }
            return View();
        }
        [HttpGet]
        public ActionResult AddDocument()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            ViewBag.msg = null;
            return View();
        }
        [HttpPost]
        public ActionResult AddDocument(string docTitle, DateTime pdate,int publisherId, int libId, int copyno, string position)
        {
            string msg = "Some error occurred. Document is not added";
            msg = dal.AddDocument(docTitle, pdate.Date, publisherId, libId, copyno, position);
            ViewBag.msg = msg;
            return View();
        }
        [HttpGet]
        public ActionResult Search()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            ViewBag.msg = null;
            return View();
        }
        [HttpPost]
        public ActionResult Search(int docId, int libId, int copyno)
        {
                string msg = dal.CheckDocumentStatus(docId, libId, copyno);
                ViewBag.msg = "Status: " + msg;
                return View();
        }
        [HttpGet]
        public ActionResult AddReader()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            ViewBag.msg = null;
            return View();
        }
        [HttpPost]
        public ActionResult AddReader(string readerName, string type, string phoneNo, string address)
        {
            string pattern = @"^\+1(\d){10}$";
            Match match = Regex.Match(phoneNo, pattern);
            if (readerName.Length > 20)
            {
                ViewBag.msg = "Reader name is too long";
                return View();
            }
            else if (match.Success == false)
            {
                ViewBag.msg = "Invalid phone no. Besure to enter "+1"";
                return View();
            }
            else if (address.Length > 50)
            {
                ViewBag.msg = "Too long address";
                return View();
            }
            else
            {
                ViewBag.msg = dal.AddReader(readerName, type, phoneNo, address);
                return View();
            }
        }
        [HttpGet]
        public ActionResult BranchInfo()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            ViewBag.msg = null;
            return View();
        }
        [HttpPost]
        public ActionResult BranchInfo(int? libId)
        {
            DataTable dt = new DataTable();
            dt = dal.GetBranchesInfo(libId);
            return View("Branch_Information", dt);
        }
        [HttpGet]
        public ActionResult FrequentBorrowers()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            return View("Frequent_Borrowers");
        }
        [HttpPost]
        public ActionResult FrequentBorrowers(int libId)
        {
            DataTable dt = new DataTable();
            dt = dal.GetTopBorrowers(libId);
            return View(dt);
        }
        [HttpGet]
        public ActionResult MostBorrowedBooks()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            return View();
        }
        [HttpPost]
        public ActionResult MostBorrowedBooks(int libId)
        {
            DataTable dt = new DataTable();
            dt = dal.GetTopBorrowedBooks(libId);
            return View("Most_Borrowed_Books",dt);
        }
        [HttpGet]
        public ActionResult MostPopularbooks()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            return View();
        }
        [HttpPost]
        public ActionResult MostPopularbooks(int year)
        {
            DataTable dt = new DataTable();
            dt = dal.GetMostPopularBooks(year);
            ViewBag.year = year;
            return View("Most_Popular_Books", dt);
        }
        [HttpGet]
        public ActionResult AverageFine()
        {
            if (HttpContext.Session["adminId"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            DataTable dt = dal.GetAvgFinePerReader();
            return View(dt);
        }
        public ActionResult Quit()
        {
            HttpContext.Session["adminId"] = null;
            return RedirectToAction("Index", "Home");
        }
            }
}
```

- **UserController:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using CityLibrary.Models;
using System.Data;

namespace CityLibrary.Controllers
{
    public class UserController : Controller
    {
        private CityLibraryDAL dal = new CityLibraryDAL();
        [HttpGet]
        public ActionResult Index()
        {
            return View();
        }
        [HttpPost]
        public ActionResult Index(int? cardno)
        {

            if (cardno != null)
            {
                int readerId = cardno??default(int);
                String readerName = dal.GetReaderById(readerId);
                if (!String.IsNullOrEmpty(readerName))
                {
                    HttpContext.Session["readerId"] = readerId;
                    HttpContext.Session["readerName"] = readerName;
                    HttpContext.Session["adminId"] = null;
                    return RedirectToAction("Main");
                }
                else
                {
                    return View();
                }
            }
            else
            {
                return View();
            }
        }
        [HttpGet]
        public ActionResult Main()
        {
            if (HttpContext.Session["readerName"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return View();
            }
        }
        [HttpGet]
        public ActionResult Search()
        {
            if (HttpContext.Session["readerName"] == null)
            {
                return RedirectToAction("Index", "Home");
            }
            else
            {
                return View();
            }
        }
        [HttpPost]
        public ActionResult Search(int? docId, string docTitle, string pubName)
        {
            DataTable dt = new DataTable();
            dt = dal.Search(docId, docTitle, pubName);
            if (dt != null)
            {
                return View("Search_Result",dt);
            }
```

```
○           else
○           {
○               return RedirectToAction("Search","User");
○           }
○       }
○       [HttpGet]
○       public ActionResult DocumentCheckout()
○       {
○           if (HttpContext.Session["readerName"] == null)
○           {
○               return RedirectToAction("Index", "Home");
○           }
○           ViewBag.msg = null;
○           return View("Document_Checkout");
○       }
○       [HttpPost]
○       public ActionResult DocumentCheckout(int docId, int copyNo, int libId)
○       {
○           int result = dal.DocumentCheckout(Convert.ToInt32(HttpContext.Session["readerId"]), docId, copyNo, libId);
○           if (result > 0)
○           {
○               ViewBag.msg = "Document checkout: Success Borrow-Transaction no: " + result;
○               return View("Document_Checkout");
○           }
○           else if (result == 0)
○           {
○               ViewBag.msg = "Document has been borrowed or reserved by another reader";
○               return View("Document_Checkout");
○           }
○           else if(result == -1)
○           {
○               ViewBag.msg = "Cannot find requested document";
○               return View("Document_Checkout");
○           }
○           else
○           {
○               ViewBag.msg = "Some Error Occurred";
○               return View("Document_Checkout");
○           }
○       }
```

```
o       [HttpGet]
o       public ActionResult DocumentReturn()
o       {
o           if (HttpContext.Session["readerName"] == null)
o           {
o               return RedirectToAction("Index", "Home");
o           }
o           return View();
o       }
o       [HttpPost]
o       public ActionResult DocumentReturn(int docId, int copyNo, int libId)
o       {
o           int result = dal.DocumentReturn(Convert.ToInt32(HttpContext.Session["readerId"]), docId, copyNo, libId);
o           if (result == 1)
o           {
o               ViewBag.msg = "Document return: Success";
o               return View("Document_Checkout");
o           }
o           else if (result == 0)
o           {
o               ViewBag.msg = "Error occurred: Found multiple records.";
o               return View("Document_Checkout");
o           }
o           else if (result == -1)
o           {
o               ViewBag.msg = "Error: You did not borrowed this document.";
o               return View("Document_Checkout");
o           }
o           else
o           {
o               ViewBag.msg = "Some Error Occurred";
o               return View("Document_Checkout");
o           }
o       }
o       [HttpGet]
o       public ActionResult DocumentReserve()
o       {
o           if (HttpContext.Session["readerName"] == null)
o           {
o               return RedirectToAction("Index", "Home");
o           }
o           ViewBag.msg = null;
○           return View();
○       }
○       [HttpPost]
○       public ActionResult DocDetails(string docTitle)
○       {
○           DataTable dt = new DataTable();
○           dt = dal.GetDocumentDetailsFromTitle(docTitle);
○           return View(dt);
○       }
○       [HttpPost]
○       public ActionResult DocumentReserve(int docId, int copyNo, int libId, DateTime resDate)
○       {
○           int result = dal.DocumentReserve(Convert.ToInt32(HttpContext.Session["readerId"]), docId, copyNo, libId, resDate);
○           if (result > 0)
○           {
○               ViewBag.msg = "Document checkout: Success Reservation no: " + result;
○               return View();
○           }
○           else if (result == 0)
○           {
○               ViewBag.msg = "You have exceeded reservation limit: 10";
○               return View();
○           }
○           else if (result == -1)
○           {
○               ViewBag.msg = "Document has been borrowed or reserved by another reader";
○               return View();
○           }
○           else if (result == -2)
○           {
○               ViewBag.msg = "Cannot find requested document";
○               return View();
○           }
○           else
○           {
○               ViewBag.msg = "Some Error Occurred";
○               return View();
○           }
○       }
```

```
o       [HttpGet]
o       public ActionResult FinesDue()
o       {
o           if (HttpContext.Session["readerName"] == null)
o           {
o               return RedirectToAction("Index", "Home");
o           }
o           ViewBag.msg = null;
o           return View();
o       }
o       [HttpPost]
o       public ActionResult FinesDue(int docId, int copyNo, int libId)
o       {
o           ViewBag.msg = dal.GetFine(Convert.ToInt32(HttpContext.Session["readerId"]), docId, copyNo, libId);
o           return View();
o       }
o       [HttpGet]
o       public ActionResult ReservedDocuments()
o       {
o           if (HttpContext.Session["readerName"] == null)
o           {
o               return RedirectToAction("Index", "Home");
o           }
o           DataTable dt = new DataTable();
o           dt = dal.ReservedDocuments(Convert.ToInt32(HttpContext.Session["readerId"]));
o           return View(dt);
o       }
o       [HttpGet]
o       public ActionResult DocumentsFromPublisher()
o       {
o           if (HttpContext.Session["readerName"] == null)
o           {
o               return RedirectToAction("Index", "Home");
o           }
o           return View();
o       }
o       [HttpPost]
o       public ActionResult DocumentsFromPublisher(string pubName)
o       {
o           DataTable dt = new DataTable();
o           dt = dal.GetDocuments(pubName);
```

```
            return View("Documents_From_Publisher",dt);
        }
        public ActionResult Quit()
        {
            HttpContext.Session["readerName"] = null;
            HttpContext.Session["readerId"] = null;
            return RedirectToAction("Index", "Home");
        }
            }
}
```

- **HomeController:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace CityLibrary.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/
        public ActionResult Index()
        {
            HttpContext.Session["adminId"] = null;
            HttpContext.Session["readerName"] = null;
            HttpContext.Session["readerId"] = null;
            return View();
        }
            }
}
```

- **CityLibraryAdminDAL:**

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Text.RegularExpressions;
using System.Linq;
using System.Web;

namespace CityLibrary.Models
```

```
{
    public class CityLibraryAdminDAL
    {
        static string connectionString = ConfigurationManager.ConnectionStrings["conString"].ToString();
        SqlConnection con = new SqlConnection(connectionString);

        public string AddDocument(string docTitle, DateTime pdate, int publisherId, int libId, int copyno, string position)
        {
            string status = "Some error occured";
            Match result = Regex.Match(position, @"^\d{3}[a-zA-z]{1}\d{2}$");
            if (result.Success)
            {
                SqlCommand cmd = new SqlCommand("spAddDocument", con);
                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.AddWithValue("@docTitle", docTitle);
                cmd.Parameters.AddWithValue("@pdate", pdate.Date);
                cmd.Parameters.AddWithValue("@pubid", publisherId);
                cmd.Parameters.AddWithValue("@libid", libId);
                cmd.Parameters.AddWithValue("@copyno", copyno);
                cmd.Parameters.AddWithValue("@position", position);
                var outparam = cmd.Parameters.AddWithValue("@status", DbType.Int32);
                outparam.Direction = ParameterDirection.Output;
                try
                {
                    con.Open();
                    cmd.ExecuteNonQuery();
                    int value = Convert.ToInt32(outparam.Value);
                    if (value == 1)
                    {
                        status = "Document added successfully";
                    }
                    else if (value == 0)
                    {
                        status = "Publisher doesn't exist";
                    }
                    else if (value == -1)
                    {
                        status = "Branch doesn't exist";
                    }
                    else
                    {
                        status = "Some error occurred";
                    }
                }
                catch (Exception)
                {
                    status = "Some error occured";
                }
                finally
                {
                    con.Close();
                }
            }
            else
            {
                status = "Invalid value in the \'position\' field";
            }

            return status;
        }

        public string CheckDocumentStatus(int docId, int libId, int copyno)
        {
            string status = "Something went wrong";
            int value = 0;
            SqlCommand cmd = new SqlCommand("spCheckDocStatus", con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@docId", docId);
            cmd.Parameters.AddWithValue("@libId", libId);
            cmd.Parameters.AddWithValue("@copyno", copyno);
            var outparam = cmd.Parameters.AddWithValue("@status", DbType.Int32);
            outparam.Direction = ParameterDirection.Output;
            try
            {
                con.Open();
                cmd.ExecuteNonQuery();
                value = (int)outparam.Value;
                if (value == 1)
                {
                    status = "Available";
                }
                else if (value == 0)
                {
                    status = "Document does not exist";
                }
                else if (value == -1)
                {
                    status = "Borrowed";
                }
                else if (value == -2)
                {
                    status = "Reserved";
                }

            }
            catch (Exception)
            {
                status = "Something went wrong";
            }
            finally
            {
                con.Close();
            }
            return status;
        }

        public string AddReader(string readerName, string type, string phoneNo, string address)
        {
            string status = "Error Can't add reader";
            SqlCommand cmd = new SqlCommand("spAddReader",con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@readerName", readerName);
            cmd.Parameters.AddWithValue("@type", type);
            cmd.Parameters.AddWithValue("@phoneno", phoneNo);
            cmd.Parameters.AddWithValue("@address", address);
            try
            {
                con.Open();
                cmd.ExecuteNonQuery();
                status = "Successfully added";
            }
            catch (Exception)
            {
                status = "Error cannot add reader";
            }
            finally
            {
```

```
o               con.Close();
o           }
o           return status;
o       }
o
o       public DataTable GetBranchesInfo(int? libId)
o       {
o           DataTable dt = new DataTable();
o           SqlCommand cmd = new SqlCommand();
o           if (libId == null)
o           {
o               cmd.CommandText = "Select Name, Location from Branch";
o           }
o           else
o           {
o               cmd.CommandText = "Select Name, Location from Branch Where
  Libid=@libId";
o               cmd.Parameters.AddWithValue("@libId", libId);
o           }
o           cmd.Connection = con;
o           try
o           {
o               SqlDataAdapter adapter = new SqlDataAdapter(cmd);
o               con.Open();
o               adapter.Fill(dt);
o           }
o           catch (Exception)
o           {
o
o           }
o           return dt;
o       }
o
o       public DataTable GetTopBorrowers(int libId)
o       {
o           DataTable dt = new DataTable();
o           SqlCommand cmd = new SqlCommand("SELECT TOP(10)
  ReaderName,COUNT(BorNumber) AS Books_Borrowed from Borrows B inner
  join Reader R on B.ReaderId=R.Reader_id where B.Libid=@libId GROUP BY
  ReaderId, ReaderName ORDER BY COUNT(BorNumber) desc", con);
o           cmd.Parameters.AddWithValue("@libId", libId);
o           SqlDataAdapter adapter = new SqlDataAdapter(cmd);
o           try
o           {
```

```
                con.Open();
                adapter.Fill(dt);
            }
            catch (Exception)
            {

            }
            return dt;
        }

        public DataTable GetTopBorrowedBooks(int libId)
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = new SqlCommand("SELECT TOP(10) Title,COUNT(BorNumber) AS 'Count' from Borrows B inner join Book BO on B.DocID=BO.DocId inner join Document D on D.DocId = BO.DocID WHERE B.Libid=@libId GROUP BY B.DocID, Title ORDER BY COUNT(BorNumber) desc", con);
            cmd.Parameters.AddWithValue("@libId",libId);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            try
            {
                con.Open();
                adapter.Fill(dt);
            }
            catch (Exception)
            {

            }
            return dt;
        }

        public DataTable GetMostPopularBooks(int year)
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = new SqlCommand("SELECT TOP(10) Title,COUNT(B.BorNumber) AS 'Count' from Borrows B inner join Book BO on B.DocID=BO.DocId inner join Document D on D.DocId = BO.DocID inner join BOR_Transaction BOR on BOR.BorNumber=B.BorNumber where year(BorDateTime)=@year Group By B.DocID,Title Order by Count(B.BorNumber) desc", con);
            cmd.Parameters.AddWithValue("@year", year);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            try
            {
                con.Open();
                adapter.Fill(dt);
            }
            catch (Exception)
            {

            }
            return dt;
        }

        public DataTable GetAvgFinePerReader()
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = new SqlCommand("SELECT B.ReaderId, R.ReaderName, ABS(AVG((DATEDIFF(DAY,BorDateTime,RetDateTime)-20)*0.20)) as 'Avg Fine' from Reader R inner join Borrows B on R.Reader_id=B.ReaderId inner join BOR_Transaction BOR on B.BorNumber=BOR.BorNumber where RetDateTime is not null Group By B.Readerid,R.ReaderName", con);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            try
            {
                con.Open();
                adapter.Fill(dt);
            }
            catch (Exception)
            {

            }
            return dt;
        }
    }
}
```

- **CityLibraryDAL:**

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using System.Linq;
using System.Web;

```

```
namespace CityLibrary.Models
{
    public class CityLibraryDAL
    {
        static string connectionString = ConfigurationManager.ConnectionStrings["conString"].ToString();
        SqlConnection con = new SqlConnection(connectionString);
        public String GetReaderById(int id)
        {
            String readerName = null;
            SqlCommand cmd = new SqlCommand("SELECT ReaderName FROM Reader WHERE Reader_id=@id",con);
            cmd.Parameters.AddWithValue("@id", id);
            try
            {
                con.Open();
                readerName = cmd.ExecuteScalar().ToString();
            }
            catch (Exception)
            {
                readerName = null;
            }
            finally
            {
                cmd.Dispose();
                con.Close();
            }
            return readerName;
        }

        public DataTable Search(int? docId, string docTitle, string pubName)
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = null;
            if (docId != null && String.IsNullOrEmpty(docTitle) && String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE DocId=@docId", con);
                cmd.Parameters.AddWithValue("@docId", docId);
            }
            else if (docId == null && !String.IsNullOrEmpty(docTitle) && String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Title=@docTitle", con);
                cmd.Parameters.AddWithValue("@docTitle", docTitle);
            }
            else if (docId == null && String.IsNullOrEmpty(docTitle) && !String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Pub_Name=@pubName", con);
                cmd.Parameters.AddWithValue("@pubName", pubName);
            }
            else if (docId == null && !String.IsNullOrEmpty(docTitle) && !String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Pub_Name=@pubName AND Title=@docTitle", con);
                cmd.Parameters.AddWithValue("@pubName", pubName);
                cmd.Parameters.AddWithValue("@docTitle", docTitle);

            }
            else if (docId != null && String.IsNullOrEmpty(docTitle) && !String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Pub_Name=@pubName AND DocId=@docId", con);
                cmd.Parameters.AddWithValue("@pubName", pubName);
                cmd.Parameters.AddWithValue("@docId", docId);
            }
            else if (docId != null && !String.IsNullOrEmpty(docTitle) && !String.IsNullOrEmpty(pubName))
            {
                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Pub_Name=@pubName AND DocId=@docId AND Title=@docTitle", con);
                cmd.Parameters.AddWithValue("@pubName", pubName);
                cmd.Parameters.AddWithValue("@docId", docId);
                cmd.Parameters.AddWithValue("@docTitle", docTitle);
            }
```

```
o            else if (docId != null && !String.IsNullOrEmpty(docTitle) && String.IsNullOrEmpty(pubName))
o            {
o                cmd = new SqlCommand("SELECT DocId,Title,Pdate,Pub_Name FROM Document D inner join Publisher P ON D.PublisherID = P.Publisher_id WHERE Title=@docTitle AND DocId=@docId", con);
o                cmd.Parameters.AddWithValue("@docTitle", docTitle);
o                cmd.Parameters.AddWithValue("@docId", docId);
o            }
o            else
o            {
o                return null;
o            }
o            try
o            {
o                con.Open();
o                SqlDataAdapter adapter = new SqlDataAdapter(cmd);
o                adapter.Fill(dt);
o            }
o            catch (Exception)
o            {
o                dt = null;
o            }
o            return dt;
o        }
o
o        public int DocumentCheckout(int readerId, int docId, int copyNo, int libId)
o        {
o            int result = -2;
o            SqlCommand cmd = new SqlCommand("spCheckout", con);
o            cmd.CommandType = CommandType.StoredProcedure;
o            cmd.Parameters.AddWithValue("@readerId", readerId);
o            cmd.Parameters.AddWithValue("@docId", docId);
o            cmd.Parameters.AddWithValue("@copyNo", copyNo);
o            cmd.Parameters.AddWithValue("@libId", libId);
o            var resultParam = cmd.Parameters.Add("@result",SqlDbType.Int);
o            resultParam.Direction = ParameterDirection.Output;
o            try
o            {
o                con.Open();
o                cmd.ExecuteNonQuery();
o                result = Convert.ToInt32(resultParam.Value);
o            }
o            catch (Exception)
```

```
        {
            result = -2;
        }
        finally
        {
            con.Close();
        }
        return result;
    }

    public int DocumentReturn(int readerId, int docId, int copyNo, int libId)
    {
        int result = -2;
        SqlCommand cmd = new SqlCommand("spReturn", con);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@readerId", readerId);
        cmd.Parameters.AddWithValue("@docId", docId);
        cmd.Parameters.AddWithValue("@copyNo", copyNo);
        cmd.Parameters.AddWithValue("@libId", libId);
        var resultParam = cmd.Parameters.Add("@result", SqlDbType.Int);
        resultParam.Direction = ParameterDirection.Output;
        try
        {
            con.Open();
            cmd.ExecuteNonQuery();
            result = Convert.ToInt32(resultParam.Value);
        }
        catch (Exception)
        {
            result = -2;
        }
        finally
        {
            con.Close();
        }
        return result;
    }

    public DataTable GetDocumentDetailsFromTitle(string docTitle)
    {
        DataTable dt = new DataTable();
        SqlCommand cmd = new SqlCommand("SELECT D.DocID, Title, Pub_Name, C.LibId, Name, CopyNo from Document D inner join Copy C on
D.DocId=C.DocID inner join Publisher P on D.PublisherID=P.Publisher_id inner join Branch B ON B.Libid=C.LibId WHERE Title=@docTitle", con);
            cmd.Parameters.AddWithValue("@docTitle",docTitle);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            try
            {
                con.Open();
                adapter.Fill(dt);
            }
            catch (Exception)
            {

            }
            return dt;
        }

        public int DocumentReserve(int readerId, int docId, int copyNo, int libId, DateTime resDate)
        {
            int result = -3;
            SqlCommand cmd = new SqlCommand("spReserve", con);
            cmd.CommandType = CommandType.StoredProcedure;
            cmd.Parameters.AddWithValue("@readerId", readerId);
            cmd.Parameters.AddWithValue("@docId", docId);
            cmd.Parameters.AddWithValue("@copyNo", copyNo);
            cmd.Parameters.AddWithValue("@libId", libId);
            cmd.Parameters.AddWithValue("@resDate", resDate.Date);
            var resultParam = cmd.Parameters.Add("@result", SqlDbType.Int);
            resultParam.Direction = ParameterDirection.Output;
            try
            {
                con.Open();
                cmd.ExecuteNonQuery();
                result = Convert.ToInt32(resultParam.Value);
            }
            catch (Exception)
            {
                result = -3;
            }
            finally
            {
                con.Close();
            }
            return result;
        }

        public string GetFine(int readerId, int docId, int copyNo, int libId)
        {
            string fine = null;
            SqlCommand cmd = new SqlCommand("select MAX(BorDateTime) from BOR_Transaction BT inner join Borrows B on BT.BorNumber=B.BorNumber where ReaderId=@readerID and DocID=@docId and Copyno=@copyNo and LibId=@libId and RetDateTime is null", con);
            cmd.Parameters.AddWithValue("@readerID", readerId);
            cmd.Parameters.AddWithValue("@docId", docId);
            cmd.Parameters.AddWithValue("@copyNo", copyNo);
            cmd.Parameters.AddWithValue("@libId", libId);
            try
            {
                con.Open();
                object d = cmd.ExecuteScalar();
                if (d is DBNull)
                {
                    fine = "You either have not borrowed this book or have already returned it.";
                }
                else
                {
                    DateTime dt = (DateTime)d;
                    int days = (DateTime.Today - dt).Days;
                    if (days > 20)
                    {
                        fine = ((days - 20) * 0.20).ToString();
                    }
                    else
                    {
                        fine = "No Fine";
                    }
                }
            }
            catch (Exception e)
            {
                fine = "Some error occurred";
            }
            finally
            {
                con.Close();
            }
```

```
○               return fine;
○           }
○
○           public DataTable ReservedDocuments(int readerId)
○           {
○               DataTable dt = new DataTable();
○               SqlCommand cmd = new SqlCommand("spGetReservedDocs", con);
○               cmd.CommandType = CommandType.StoredProcedure;
○               cmd.Parameters.AddWithValue("@readerId", readerId);
○               try
○               {
○                   con.Open();
○                   SqlDataAdapter adapter = new SqlDataAdapter(cmd);
○                   adapter.Fill(dt);
○               }
○               catch (Exception)
○               {
○                   dt = null;
○               }
○               return dt;
○           }
○
○           public DataTable GetDocuments(string pubName)
○           {
○               DataTable dt = new DataTable();
○               SqlCommand cmd = new SqlCommand("select DocId,Title from Document D inner join Publisher P on D.PublisherID=P.Publisher_id where Pub_Name = @pubName", con);
○               cmd.Parameters.AddWithValue("@pubName", pubName);
○               try
○               {
○                   con.Open();
○                   SqlDataAdapter adapter = new SqlDataAdapter(cmd);
○                   adapter.Fill(dt);
○               }
○               catch (Exception)
○               {
○                   dt = null;
○               }
○               return dt;
○           }
○       }
○   }
```