LIBRARY MANAGEMENT SYSTEM - (BEGINNER LEVEL)

Creating Tables

So our first step is to create tables. Let us create a database as db_LibraryManagement and then create all the required tables inside it.

```
CREATE PROC dbo.LibraryManagementSystemProcedure
AS CREATE DATABASE db_LibraryManagement
GO
CREATE TABLE table_publisher (
PublisherName VARCHAR(50) PRIMARY KEY NOT NULL,
PublisherAddress VARCHAR(100) NOT NULL,
PublisherPhone VARCHAR(20) NOT NULL,
);
```

Now let's create a table for a book.

```
CREATE TABLE table_book (
BookID INT PRIMARY KEY NOT NULL IDENTITY (1,1),
Book_Title VARCHAR(100) NOT NULL,
PublisherName VARCHAR(100) NOT NULL
);
```

Similarly, create a table for the library branch.

```
CREATE TABLE table_library_branch (
library_branch_BranchID INT PRIMARY KEY NOT NULL IDENTITY (
```

```
library branch BranchName VARCHAR(100) NOT NULL,
library_branch_BranchAddress VARCHAR(200) NOT NULL,
);
```

View the library branch table.

```
SELECT FROM table_library_branch
CREATE TABLE table_borrower (
CardNo INT PRIMARY KEY NOT NULL IDENTITY (100,1),
BorrowerName VARCHAR(100) NOT NULL,
BorrowerAddress VARCHAR(200) NOT NULL,
BorrowerPhone VARCHAR(50) NOT MILL,
);
```

Create a table to store the book copies.

```
CREATE TABLE table_book_copies (
book_copies CopiesID INT PRIMARY KEY NOT NULL
book_copies BookID INT NOT NULL
book_copies BranchID INT NOT NULL
book_copies No Of Copies INT NOT NULL,
);
```

Create one more table for storing book authors

```
SELECT FROM table_book_copies CREATE TABLE table_book_authors (
book_authors AuthorID INT PRIMARY KEY NOT NULL IDENTITY (1,1),
```

```
book_authors BookID INT NOT NULL CONSTRAINT fk_book_id3 FOREIGN KEY
REFERENCES table_book(book_BookID) ON UPDATE CASCADE ON DELETE
CASCADE, table_book(book_BookID) ON UPDATE CASCADE,

book authors AuthorName VARCHAR(50) NOT NULL,

);
```

```
SELECT FROM table_book_authors
```

Inserting Data

Next is having some data inserted into the tables. Let's do it now.

Table structure for table `book`

```
CREATE TABLE IF NOT EXISTS `book` (

  `isbn` char(13) NOT NULL,

  `title` varchar(80) NOT NULL,

  `author` varchar(80) NOT NULL,

  `category` varchar(80) NOT NULL,

  `price` int(4) unsigned NOT NULL,

  `copies` int(10) unsigned NOT NULL

);
```

Adding data into the table book.

```
INSERT INTO `book` (`isbn`, `title`, `author`, `category`, `price`, `copies`) VALUES

('9788654552277', 'X-Men: God Loves, Man Kills', 'Chris', 'Comics', 98, 39),

('0964161484100', 'Mike Tyson : Undisputed Truth', 'Larry Sloman, Mike Tyson',
'Sports', 654, 79),

('6901142585540', 'V for Vendetta', 'Alan Moore', 'Comics', 600, 23),

('9094996245442', 'When Breath Becomes Air', 'Paul Kalanithi', 'Medical', 500, 94),

('8653491200700', 'The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', 432, 120);
```

Structure of the table for book issue.

```
CREATE TABLE IF NOT EXISTS `book_issue` (

`issue_id` int(11) NOT NULL,

  `member` varchar(20) NOT NULL,

  `book_isbn` varchar(13) NOT NULL,

  `due_date` date NOT NULL,

  `last_reminded` date DEFAULT NULL
```

```
) ;
CREATE TRIGGER `issue_book` BEFORE INSERT ON `book_issue`

 FOR EACH ROW BEGIN

  SET NEW.due_date = DATE_ADD(CURRENT_DATE, INTERVAL 20 DAY);

    UPDATE member SET balance = balance - (SELECT price FROM book WHERE

isbn = NEW.book_isbn) WHERE username = NEW.member;

    UPDATE book SET copies = copies - 1 WHERE isbn = NEW.book_isbn;

    DELETE FROM pending_book_requests WHERE member = NEW.member AND
book_isbn = NEW.book_isbn;

END
CREATE TRIGGER `return_book` BEFORE DELETE ON `book_issue`

 FOR EACH ROW BEGIN

    UPDATE member SET balance = balance + (SELECT price FROM book WHERE
isbn = OLD.book_isbn) WHERE username = OLD.member;

    UPDATE book SET copies = copies + 1 WHERE isbn = OLD.book_isbn;

END
```

Structure of librarian table.

```
CREATE TABLE IF NOT EXISTS `librarian` (

`id` int(11) NOT NULL,

  `username` varchar(20) NOT NULL,

  `password` char(40) NOT NULL

) ;
```

Adding details of the librarian.

```
INSERT INTO `librarian` (`id`, `username`, `password`) VALUES

(1, 'Vani', 'xthds97@3h$yfc*jrk0%dfg
```

Structure of table for the member.

```
CREATE TABLE IF NOT EXISTS `member` (

`id` int(11) NOT NULL,

  `username` varchar(20) NOT NULL,

  `password` char(40) NOT NULL,

  `name` varchar(80) NOT NULL,
```

```
  `email` varchar(80) NOT NULL,

  `balance` int(4) NOT NULL

);
CREATE TRIGGER `add_member` AFTER INSERT ON `member`

 FOR EACH ROW DELETE FROM pending_registrations WHERE username =
NEW.username

CREATE TRIGGER `remove_member` AFTER DELETE ON `member`

 FOR EACH ROW DELETE FROM pending_book_requests WHERE member =
OLD.username
```

Structure of table for pending book requests

```
CREATE TABLE IF NOT EXISTS `pending_book_requests` (

`request_id` int(11) NOT NULL,

  `member` varchar(20) NOT NULL,

  `book_isbn` varchar(13) NOT NULL,

  `time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
```

```
);
```

Structure of table for pending registrations.

```
CREATE TABLE IF NOT EXISTS `pending_registrations` (

  `username` varchar(30) NOT NULL,

  `password` char(20) NOT NULL,

  `name` varchar(40) NOT NULL,

  `email` varchar(20) NOT NULL,

  `balance` int(10),

  `time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP

);
```

Add data for pending registrations table.

```
INSERT INTO `pending_registrations`

(`username`, `password`, `name`, `email`, `balance`, `time`)
```

```
VALUES

('Robin200', '7t6hg$56y^', 'Robin', 'robin@gmail.com', 200, '2021-03-21 08:59:00'),

('Aadhya100', 'Ujgf(76G5$#f@df', 'Aadhya', 'aadhya100@gmail.com', 1500, '2021-03-21
2:14:53');
```

Now let’s add primary keys to these tables.

```
ALTER TABLE `book`

ADD PRIMARY KEY (`isbn`);

ALTER TABLE `book_issue`

ADD PRIMARY KEY (`issue_id`);

ALTER TABLE `librarian`

ADD PRIMARY KEY (`id`), ADD UNIQUE KEY `username` (`username`);

ALTER TABLE `member`

ADD PRIMARY KEY (`id`), ADD UNIQUE KEY `username` (`username`), ADD
UNIQUE KEY `email` (`email`);

ALTER TABLE `pending_book_requests`

ADD PRIMARY KEY (`request_id`);
```

```
ALTER TABLE `pending_registrations`

ADD PRIMARY KEY (`username`);
```

Testing

Now it's time to test our database. The below code is to test whether we are getting the correct output. We are getting to know the number of books named The Lost Tribe present in the library Sharpstown.

```
CREATE PROC dbo.bookCopiesAtAllSharpstown
AS
SELECT copies.book_copies_BranchID AS [Branch ID],
branch.library_branch_BranchName AS [Branch Name],
        copies.book_copies_No_Of_Copies AS [Number of Copies],
        book.book_Title AS [Book Title]
        FROM table_book_copies AS copies
                INNER JOIN table_book AS book ON copies.book_copies_BookID
= book.book_BookID
                INNER JOIN table_library_branch AS branch ON
book_copies_BranchID = branch.library_branch_BranchID
        WHERE book.book_Title = @bookTitle AND
branch.library_branch_BranchName = @branchName
GO
EXEC dbo.bookCopiesAtAllSharpstown


(@bookTitle varchar(70) = 'The Lost Tribe', @branchName varchar(70) = 'Sharpstown')
);
```