Sri Lanka Institute of Information Technology

IE2042: Database Management Systems for Security
Year 2, Semester 1

Group Assignment
Group No: 34
MLB_WE_01_34

Members Details

Registration Number	Name	Contact Number
IT 23 2095 34	NETHMIKA A.A.I	+94 75 970 3538
IT 23 2272 86	PEIRIS W.S.S.N	+94 70 488 4193
IT 23 2283 44	DAHANAYAKE V.T	+94 71 662 5959
IT 23 2419 78	S.D.W.GUNARATNE	+94 78 362 7788

Submission Date: 10/15/2024

1. Declaration and Group Details

We hereby declare that this Assignment is our original work, and it has not been copied or submitted in whole or part for any other module or assessment at SLIIT or any other university/ institute Due acknowledgement has been given to all the sources used. We hereby confirm that all those incorporated in our group have shared and contributed collectively to this Assignment in the research, formulation, and completion of our work by sharing the work equitably and accepting responsibility for the contribution made.

Branch: Malabe
Group No: 34
Batch: Weekend_ CS

Table Of Content

1. Introduction

Data management and security are the leading concerns in the modern technological world for the working of an organization. The present assignment is, therefore, focused and aims at designing and developing a robust database security solution using DBMS for purposes related to a university's library system. The system should be able to manage various resources, including physical books, journals, and digital media, for users like students, faculty, and staff.

It will thus enable us to show how to design a database concerned with real-world applications by considering requirements that concern data integrity, security, and efficiency. Additionally, it will outline problems related to entity relationship modeling, normalization, query optimization, and security implementation relevant to the protection of sensitive data and to assure high performance for database systems.

The project will further debate two major database vulnerabilities, their potential impacts, and the necessary countermeasures. A holistic approach will be pursued in making sure that a functional solution is also secure, and performance tuned.

2. Scenario

The university library system is a comprehensive platform designed to manage a wide array of resources, users, and administrative functions. The library houses various items, including books, journals, and digital media, each with unique identifiers and characteristics. For instance, books have attributes such as ISBN, author, and genre, while journals are identified by ISSN, volume, and issue. Digital media, on the other hand, are characterized by their format and size. Additionally, each item is categorized by broader categories such as Fiction, Non-Fiction, Science, and Humanities, and can be associated with one or more academic subjects. Items are also has a name and a publisher.

The library serves a diverse user base comprising students, faculty, staff, and librarians. Each user type has specific attributes: students have a major and year of study, faculty members belong to a department and hold a title, staff members have designated positions and offices, and librarians are either a Manager or a assistant. All users share common attributes like name, email, and phone number. Librarians play a crucial role in managing library operations, assisting users, and overseeing transactions.

Each item in the library can have multiple copies. For example, there might be several copies of the same book or journal issue. Each copy has specific attributes such as CopyID, Condition, and Location within the library or across different branches. The library also operates across multiple branches, each managed by a branch manager who is typically a senior librarian. The system tracks the distribution of items across these branches.

Users interact with the library system primarily through borrowing copy of items. When a user borrows an item, a transaction is recorded, capturing details such as the borrow date due date. These transactions link users to the specific copies of items they borrow ensuring that the system can track the movement and availability of resources. Additionally, users can provide feedback on the items they interact with through reviews, which include ratings and comments that help other users make informed decisions.

The library has a fine system in place. If a user borrows a book and does not return it or extend it within the specified time period (decided by the item type), a fine is levied on the borrower. There can be multiple fines on the same item depending on the delay. Fines are associated with specific borrowing transactions, and the system tracks the status of each fine (paid or unpaid).

The system also supports user feedback through a review system, where users can rate and comment on items they have used. These reviews help other users in making informed decisions and contribute to the library's understanding of user preferences.

Finally, the library system sends notifications that are created by Library Manager to users regarding important updates, such as due date reminders, reservation availability, and overdue items. These notifications ensure that users stay informed and can manage their interactions with the library efficiently.

3. Assumption

1. Return can have many fines, because fine can be fines can be collected from, book return delays and poor book condition.

2. Many messages deliver to many users.

3. Borrowing criteria and fine calculating criteria are same for all users (staff &student)

4. There are only 4 user types in the library system. There are student, Faculty member, staff, Librarian.

5. Staff represents without library officers. (This system is developed according to the dynamic of the university.)

6. Every Item have multiple copies.

8.Non academic staff except librarian are represented by staff.

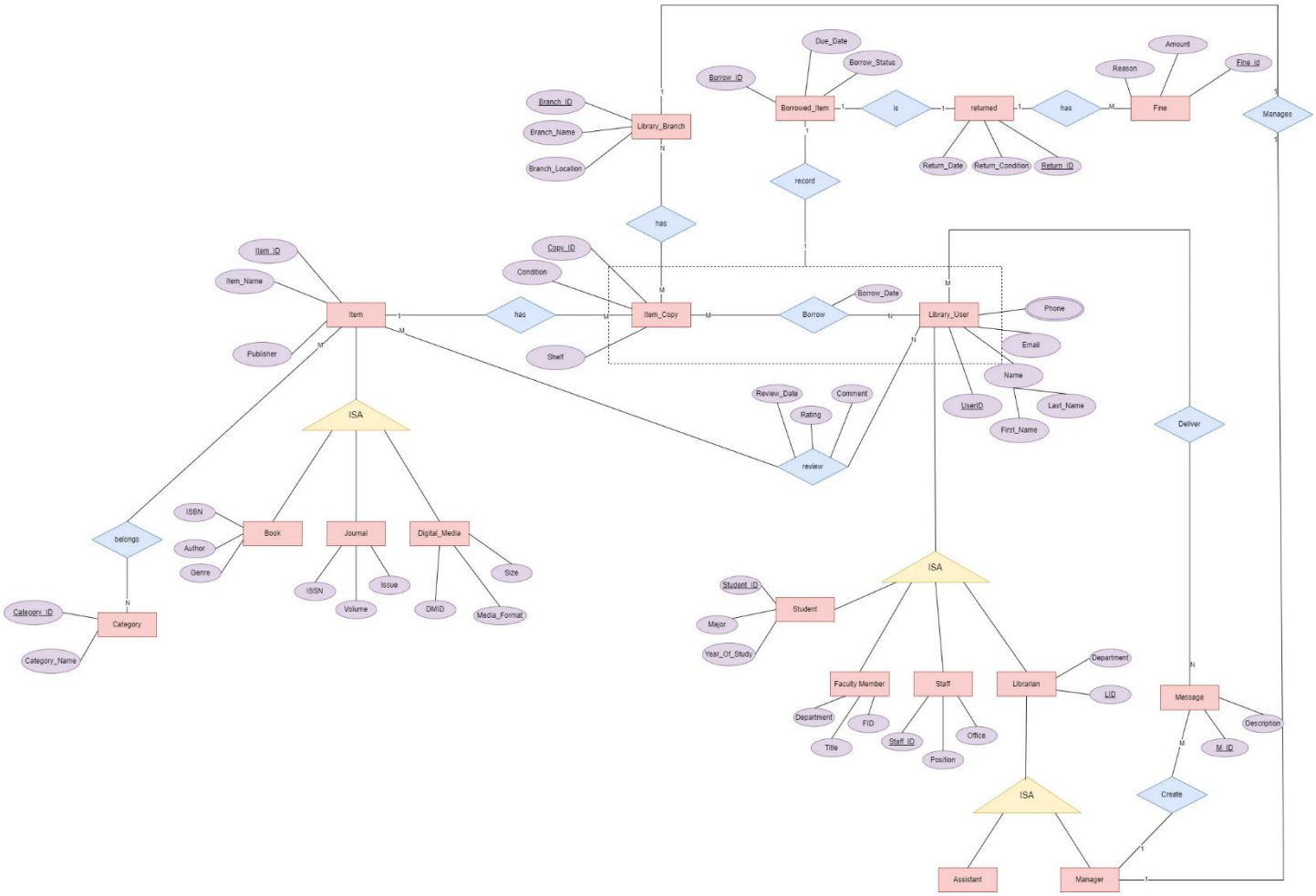
9.Assistent and Manager covers all the type of Librarian.

10.Item_copy cannot exist without Item.

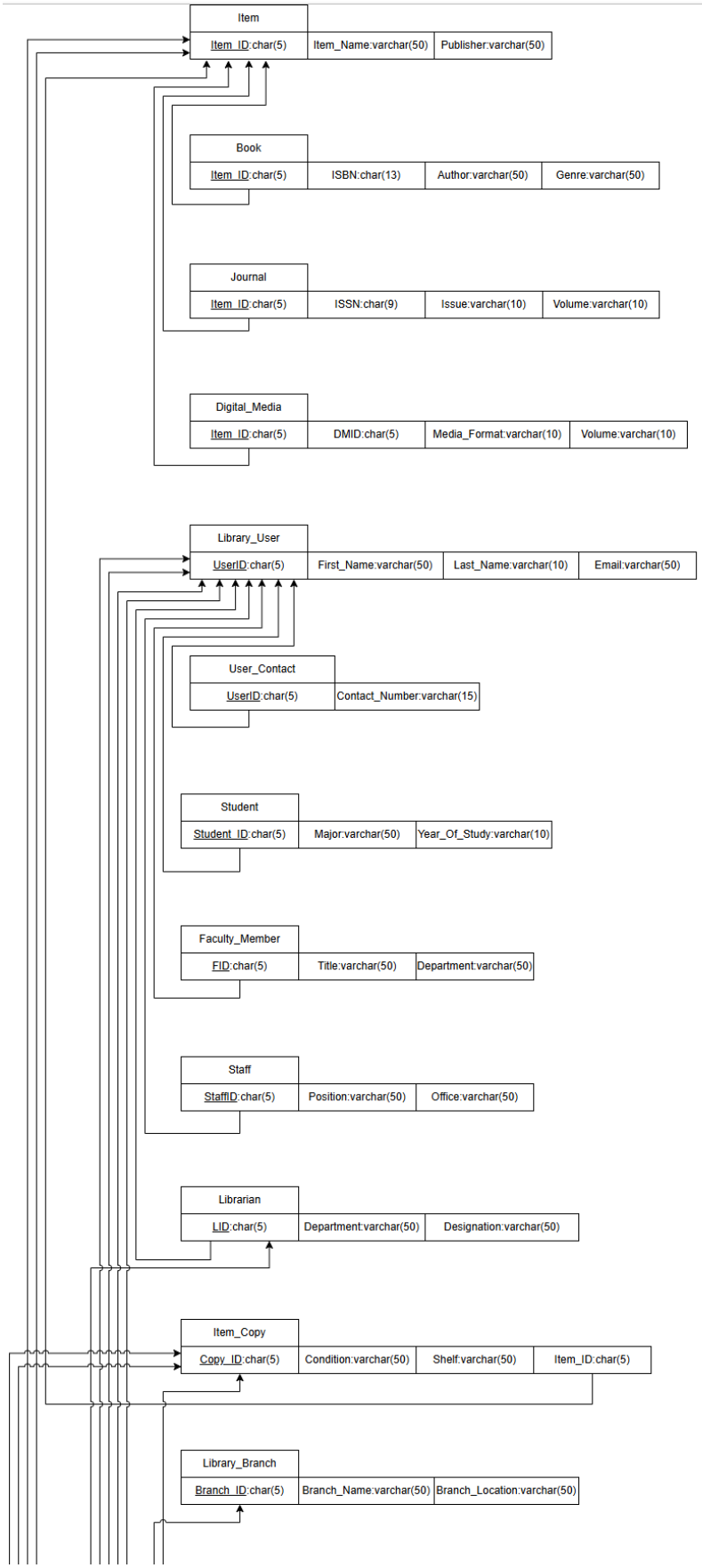
11.One item has many copies, and one copy belongs to 1 item. **

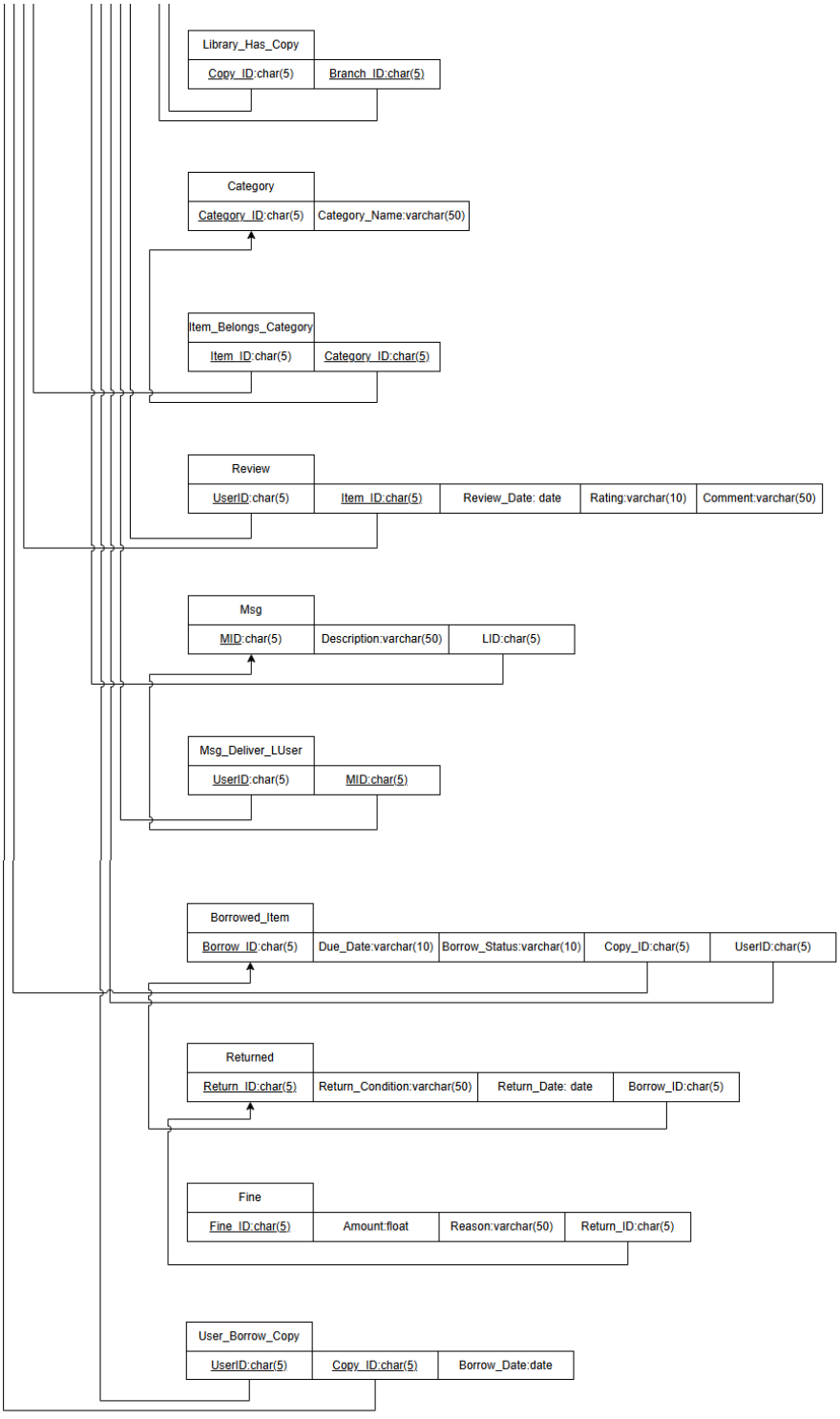
12.Only the manager can create and send messages.

4. ER Diagram

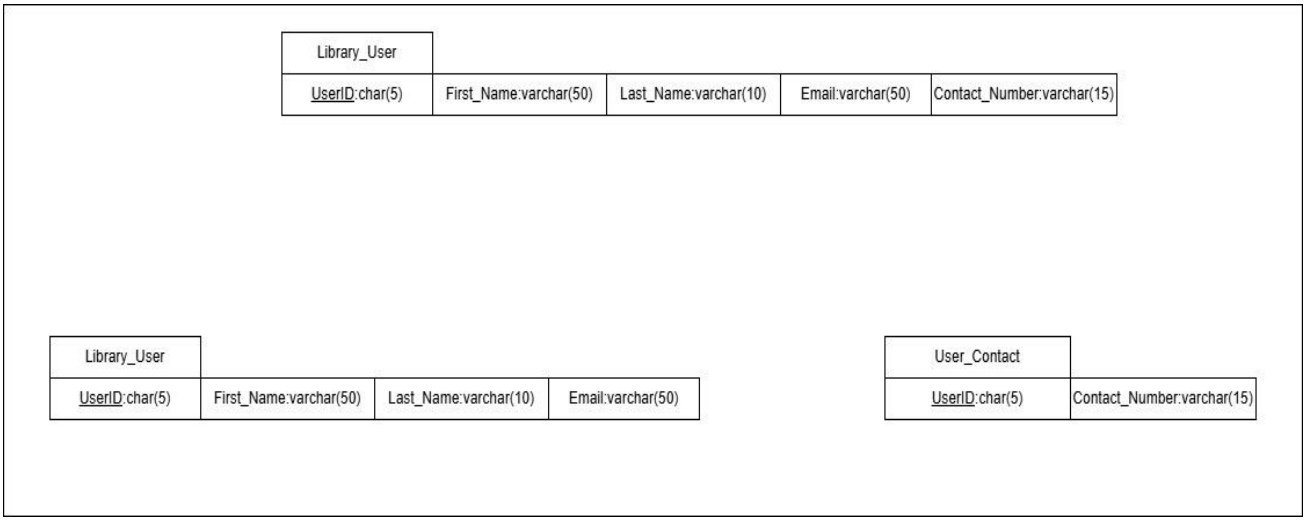

Due_Date
Borrow_Status
Borrow_ID
Borrowed_Item
is
returned
has
Fine
Amount
Reason
Fine_id
Manages
Branch_ID
Branch_Name
Branch_Location
Library_Branch
Return_Date
Return_Condition
Return_ID
record
has
Item_ID
Item_Name
Item
Publisher
Copy_ID
Condition
has
Item_Copy
Shelf
Borrow
Borrow_Date
Library_User
Phone
Email
Name
UserID
Last_Name
First_Name
Review_Date
Comment
Rating
review
ISA
Deliver
belongs
Category_ID
Category
Category_Name
ISBN
Author
Genre
Book
Journal
ISSN
Volume
Issue
Digital_Media
DMID
Media_Format
Size
ISA
Student_ID
Student
Major
Year_Of_Study
Faculty Member
Department
FID
Title
Staff
Staff_ID
Office
Position
Librarian
Department
LID
Message
Description
M_ID
Create
ISA
Assistant
Manager

5. Mapping


Item
Item_ID:char(5)
Item_Name:varchar(50)
Publisher:varchar(50)
Book
Item_ID:char(5)
ISBN:char(13)
Author:varchar(50)
Genre:varchar(50)
Journal
Item_ID:char(5)
ISSN:char(9)
Issue:varchar(10)
Volume:varchar(10)
Digital_Media
Item_ID:char(5)
DMID:char(5)
Media_Format:varchar(10)
Volume:varchar(10)
Library_User
UserID:char(5)
First_Name:varchar(50)
Last_Name:varchar(10)
Email:varchar(50)
User_Contact
UserID:char(5)
Contact_Number:varchar(15)
Student
Student_ID:char(5)
Major:varchar(50)
Year_Of_Study:varchar(10)
Faculty_Member
FID:char(5)
Title:varchar(50)
Department:varchar(50)
Staff
StaffID:char(5)
Position:varchar(50)
Office:varchar(50)
Librarian
LID:char(5)
Department:varchar(50)
Designation:varchar(50)
Item_Copy
Copy_ID:char(5)
Condition:varchar(50)
Shelf:varchar(50)
Item_ID:char(5)
Library_Branch
Branch_ID:char(5)
Branch_Name:varchar(50)
Branch_Location:varchar(50)


Library_Has_Copy
Copy_ID:char(5)
Branch_ID:char(5)
Category
Category_ID:char(5)
Category_Name:varchar(50)
Item_Belongs_Category
Item_ID:char(5)
Category_ID:char(5)
Review
UserID:char(5)
Item_ID:char(5)
Review_Date: date
Rating:varchar(10)
Comment:varchar(50)
Msg
MID:char(5)
Description:varchar(50)
LID:char(5)
Msg_Deliver_LUser
UserID:char(5)
MID:char(5)
Borrowed_Item
Borrow_ID:char(5)
Due_Date:varchar(10)
Borrow_Status:varchar(10)
Copy_ID:char(5)
UserID:char(5)
Returned
Return_ID:char(5)
Return_Condition:varchar(50)
Return_Date: date
Borrow_ID:char(5)
Fine
Fine_ID:char(5)
Amount:float
Reason:varchar(50)
Return_ID:char(5)
User_Borrow_Copy
UserID:char(5)
Copy_ID:char(5)
Borrow_Date:date

6. Normalization



7. SQL Query

7.1.Create Tables

```
CREATE TABLE Item
(
    Item_ID CHAR(5) NOT NULL,
    Item_Name VARCHAR(50) NOT NULL,
    Publisher VARCHAR(50) NOT NULL,

    CONSTRAINT Item_PK PRIMARY KEY(Item_ID)
);

CREATE TABLE Book
(
    Item_ID CHAR(5) NOT NULL,
    ISBN CHAR(13) NOT NULL,
    Author VARCHAR(50) NOT NULL,
    Genre VARCHAR(50) NOT NULL,

    CONSTRAINT Book_PK PRIMARY KEY(Item_ID),
    CONSTRAINT Book_FK FOREIGN KEY(Item_ID)
REFERENCES Item(Item_ID)
);
```

```
CREATE TABLE Journal
(
    Item_ID CHAR(5) NOT NULL,
    ISSN CHAR(9) NOT NULL,
    Issue VARCHAR(10) NOT NULL,
    Volume VARCHAR(10) NOT NULL,

    CONSTRAINT Journal_PK PRIMARY KEY(Item_ID),
    CONSTRAINT Journal_FK FOREIGN KEY(Item_ID)
REFERENCES Item(Item_ID)
);

CREATE TABLE Digital_Media
(
    Item_ID CHAR(5) NOT NULL,
    DMID CHAR(5) NOT NULL,
    Media_Format VARCHAR(10) NOT NULL,
    Volume VARCHAR(10) NOT NULL,

    CONSTRAINT Digital_Media_PK PRIMARY
KEY(Item_ID),
    CONSTRAINT Digital_Media_FK FOREIGN
KEY(Item_ID) REFERENCES Item(Item_ID)
);

CREATE TABLE Library_User
(
    UserID CHAR(5) NOT NULL,
    First_Name VARCHAR(50) NOT NULL,
```

```
    Last_Name VARCHAR(50) NOT NULL,
    Email VARCHAR(50) NOT NULL,

    CONSTRAINT Library_User_PK PRIMARY
KEY(UserID),
);

CREATE TABLE User_Contact
(
    UserID CHAR(5) NOT NULL,
    Contact_Number VARCHAR(15) NOT NULL,

    CONSTRAINT User_Contact_PK PRIMARY
KEY(UserID),
    CONSTRAINT User_Contact_FK FOREIGN
KEY(UserID) REFERENCES Library_User(UserID)
);

CREATE TABLE Student
(
    Student_ID char (5) NOT NULL,
    Major varchar (50) NOT NULL,
    Year_Of_Study varchar (10) NOT NULL,

    CONSTRAINT Student_PK PRIMARY
KEY(Student_ID),
    CONSTRAINT Student_FK FOREIGN KEY(Student_ID)
REFERENCES Library_User(UserID)
```

```
);

CREATE TABLE Faculty_Member
(
    FID char (5) NOT NULL,
    Title varchar (50) NOT NULL,
    Department varchar (50) NOT NULL,

    CONSTRAINT Faculty_Member_PK PRIMARY
KEY(FID),
    CONSTRAINT Faculty_Member_FK FOREIGN KEY(FID)
REFERENCES Library_User(UserID)

);

CREATE TABLE Staff
(
    StaffID char (5) NOT NULL,
    Position varchar (50) NOT NULL,
    Office varchar (50) NOT NULL,

    CONSTRAINT Staff_PK PRIMARY KEY(StaffID),
    CONSTRAINT Staff_FK FOREIGN KEY(StaffID)
REFERENCES Library_User(UserID)

);

CREATE TABLE Librarian
(
```

```
    LID char (5) NOT NULL,
    Department varchar (50) NOT NULL,
    Designation varchar (50) NOT NULL,

    CONSTRAINT Librarian_PK PRIMARY KEY(LID),
    CONSTRAINT Librarian_FK FOREIGN KEY(LID)
REFERENCES Library_User(UserID)

);

CREATE TABLE Item_Copy
(
    Copy_ID char (5) NOT NULL,
    Condition varchar (50) NOT NULL,
    Shelf varchar (50) NOT NULL,
    Item_ID char (5) NOT NULL,

    CONSTRAINT Item_Copy_PK PRIMARY KEY(Copy_ID),
    CONSTRAINT Item_Copy_FK FOREIGN KEY(Item_ID)
REFERENCES Item(Item_ID)

);

CREATE TABLE Library_Branch
(
    Branch_ID char (5) NOT NULL,
    Branch_Name varchar (50) NOT NULL,
    Branch_Location varchar (50) NOT NULL,
```

```
    CONSTRAINT Library_Branch_PK PRIMARY
KEY(Branch_ID)

);

CREATE TABLE Library_Has_Copy
(
    Copy_ID char (5) NOT NULL,
    Branch_ID char (5) NOT NULL,

    CONSTRAINT Library_Has_Copy_PK PRIMARY
KEY(Copy_ID, Branch_ID),
    CONSTRAINT Library_Has_Copy_FK1 FOREIGN
KEY(Copy_ID) REFERENCES Item_Copy(Copy_ID),
    CONSTRAINT Library_Has_Copy_FK2 FOREIGN
KEY(Branch_ID) REFERENCES
Library_Branch(Branch_ID)
);

CREATE TABLE Category
(
    Category_ID char (5) NOT NULL,
    Category_Name varchar (50) NOT NULL,

    CONSTRAINT Category_PK PRIMARY
KEY(Category_ID)

);
```

```
CREATE TABLE Item_Belongs_Category
(
    Item_ID char (5) NOT NULL,
    Category_ID char (5) NOT NULL,

    CONSTRAINT Item_Belongs_Category_PK PRIMARY
KEY(Item_ID, Category_ID),
    CONSTRAINT Item_Belongs_Category_FK1 FOREIGN
KEY(Item_ID) REFERENCES Item(Item_ID),
    CONSTRAINT Item_Belongs_Category_FK2 FOREIGN
KEY(Category_ID) REFERENCES Category(Category_ID)

);

CREATE TABLE Review
(
    UserID char (5) NOT NULL,
    Item_ID char (5) NOT NULL,
    Review_Date date NOT NULL,
    Rating varchar (10) NOT NULL,
    Comment varchar (50) NOT NULL,

    CONSTRAINT Review_PK PRIMARY KEY(UserID,
Item_ID),
    CONSTRAINT Review_FK1 FOREIGN KEY(UserID)
REFERENCES Library_User(UserID),
    CONSTRAINT Review_FK2 FOREIGN KEY(Item_ID)
REFERENCES Item(Item_ID)
```

```
);

 CREATE TABLE Msg
 (
    MID char (5) NOT NULL,
    Msg varchar(50) NOT NULL,
    LID char(5),

    CONSTRAINT Msg_PK PRIMARY KEY(MID),
    CONSTRAINT Msg_FK FOREIGN KEY(LID) REFERENCES
Librarian(LID)
);

CREATE TABLE Msg_Deliver_LUser
(
    UserID char (5) NOT NULL,
    MID char (5) NOT NULL,

    CONSTRAINT Msg_Deliver_LUser_PK PRIMARY
KEY(UserID, MID),
    CONSTRAINT Msg_Deliver_LUser_FK1 FOREIGN
KEY(UserID) REFERENCES Library_User(UserID),
    CONSTRAINT Msg_Deliver_LUser_FK2 FOREIGN
KEY(MID) REFERENCES Msg(MID)

);

CREATE TABLE Borrowed_Item
(
```

```
    Borrow_ID char(5) NOT NULL,
    Due_Date varchar(10) NOT NULL,
    Borrow_Status varchar(10) NOT NULL,
    Copy_ID char(5) NOT NULL,
    UserID char(5) NOT NULL,

    CONSTRAINT BORROW_Item_PK PRIMARY
KEY(Borrow_ID),
    CONSTRAINT BORROW_Item_FK1 FOREIGN
KEY(Copy_ID) REFERENCES Item_Copy(Copy_ID),
    CONSTRAINT BORROW_Item_FK2 FOREIGN
KEY(UserID) REFERENCES Library_User(UserID)

);

CREATE TABLE Returned
(
    Return_ID char(5) NOT NULL,
    Return_Condition varchar(50) NOT NULL,
    Return_Date date NOT NULL,
    Borrow_ID char(5),

    CONSTRAINT Returned_PK PRIMARY
KEY(Return_ID),
    CONSTRAINT Returned_FK FOREIGN KEY(Borrow_ID)
REFERENCES Borrowed_Item(Borrow_ID)

);
```

```
CREATE TABLE Fine
(
    Fine_ID char(5) NOT NULL,
    Amount float NOT NULL,
    Reason varchar(50) NOT NULL,
    Return_ID char(5) Not NULL,

    CONSTRAINT Fine_PK PRIMARY KEY(Fine_ID),
    CONSTRAINT Fine_FK FOREIGN KEY(Return_ID)
REFERENCES Returned(Return_ID)

);

CREATE TABLE User_Borrow_Copy
(
    UserID char(5) NOT NULL,
    Copy_ID char(5) NOT NULL,
    Borrow_Date date NOT NULL,

    CONSTRAINT User_Borrow_Copy_PK PRIMARY
KEY(UserID,Copy_ID),
    CONSTRAINT User_Borrow_Copy_FK1 FOREIGN
KEY(UserID) REFERENCES Library_User(UserID),
    CONSTRAINT User_Borrow_Copy_FK2 FOREIGN
KEY(Copy_ID) REFERENCES Item_Copy(Copy_ID)

);
```

```sql
/*check*/

ALTER TABLE Library_User
ADD CONSTRAINT Email_Check
CHECK (Email LIKE '%_@__%.__%');

/*check*/



ALTER TABLE User_Contact
ADD CONSTRAINT Contact_Number_Check
CHECK (PATINDEX('+94-[0-9][0-9]-[0-9][0-9][0-9]-
[0-9][0-9][0-9][0-9]', Contact_Number) = 1);
```

7.2. Insert Values

```
INSERT INTO Item (Item_ID, Item_Name, Publisher)
VALUES
('B0001', 'Book A', 'Publisher 1'),
('B0002', 'Book B', 'Publisher 2'),
('B0003', 'Book C', 'Publisher 3'),
('B0004', 'Book D', 'Publisher 3'),
('B0005', 'Book E', 'Publisher 4'),
('J0001', 'Journal A', 'Publisher 5'),
('J0002', 'Journal B', 'Publisher 6'),
('J0003', 'Journal C', 'Publisher 7'),
('J0004', 'Journal D', 'Publisher 8'),
('J0005', 'Journal E', 'Publisher 9'),
('M0001', 'Media A', 'Publisher 10'),
('M0002', 'Media B', 'Publisher 11'),
('M0003', 'Media C', 'Publisher 12'),
('M0004', 'Media D', 'Publisher 13'),
('M0005', 'Media E', 'Publisher 14');


INSERT INTO Book (Item_ID, ISBN, Author, Genre)
VALUES
('B0001', '7788130000000', 'J.R.R.Talkien',
'Fantasy'),
('B0002', '7788130000001', 'Arthur C. Clarke ',
'Science Fiction'),
('B0003', '7788130000002', 'J.R.R.Talkien',
'Fantasy'),
```

```
('B0004', '7788130000003', 'Jeff Kinney',
'Comedy'),
('B0005', '9780134685991', 'Kevin Kwan',
'Romantic');

INSERT INTO Journal (Item_ID, ISSN, Issue,
Volume) VALUES
('J0001', '1234-5678', 'Issue 1', 'Vol 1'),
('J0002', '2234-5678', 'Issue 2', 'Vol 2'),
('J0003', '3234-5678', 'Issue 3', 'Vol 3'),
('J0004', '4234-5678', 'Issue 4', 'Vol 4'),
('J0005', '5234-5678', 'Issue 5', 'Vol 5');

INSERT INTO Digital_Media (Item_ID, DMID,
Media_Format, Volume) VALUES
('M0001', 'D0001', 'Video', 'Vol 1'),
('M0002', 'D0002', 'Audio', 'Vol 2'),
('M0003', 'D0003', 'Audio', 'Vol 3'),
('M0004', 'D0004', 'Text', 'Vol 4'),
('M0005', 'D0005', 'Text', 'Vol 5');
```

```
INSERT INTO Library_User (UserID, First_Name,
Last_Name, Email) VALUES
('S0001', 'Sandul', 'Wickrama',
'sandul@example.com'),
('S0002', 'Sugreewa', 'Rajapaksha',
'sugreewa@example.com'),
('S0003', 'Vikum', 'Jayawardana',
'vikum@example.com'),
('S0004', 'Isindu', 'Jayasooriya',
'isindu@example.com'),
('S0005', 'Dimuthu', 'Dilshan',
'dimuthu@example.com'),
('F0001', 'Kasun', 'Perera',
'kasun.perera@example.com'),
('F0002', 'Nuwan', 'Silva',
'nuwan.silva@example.com'),
('F0003', 'Amaya', 'Fernando',
'amaya.fernando@example.com'),
('F0004', 'Sanduni', 'Wijesinghe',
'sanduni.wijesinghe@example.com'),
('F0005', 'Ruwan', 'Jayasinghe',
'ruwan.jayasinghe@example.com'),
('Sf001', 'Tharushi', 'Gunawardena',
'tharushi.gunawardena@example.com'),
('Sf002', 'Anura', 'Ratnayake',
'anura.ratnayake@example.com'),
('Sf003', 'Chathura', 'Dias',
'chathura.dias@example.com'),
```

```
('Sf004', 'Samantha', 'De Silva',
'samantha.desilva@example.com'),
('Sf005', 'Ishara', 'Hettiarachchi',
'ishara.hettiarachchi@example.com'),
('L0001', 'Kavinda', 'Abeysekera',
'kavinda.abeysekera@example.com'),
('L0002', 'Dilshan', 'Senanayake',
'dilshan.senanayake@example.com'),
('L0003', 'Nadeeka', 'Bandara',
'nadeeka.bandara@example.com'),
('L0004', 'Thilini', 'Rajapaksha',
'thilini.rajapaksha@example.com'),
('L0005', 'Suresh', 'Dissanayake',
'suresh.dissanayake@example.com');


INSERT INTO User_Contact (UserID, Contact_Number)
VALUES
('S0001', '+94-77-111-1117'),
('S0002', '+94-71-444-5555'),
('S0003', '+94-76-555-6666'),
('S0004', '+94-72-888-9999'),
('S0005', '+94-72-999-9999');

INSERT INTO Student (Student_ID, Major,
Year_Of_Study) VALUES
('S0001', 'Information Technology', 'First'),
('S0002', 'Information Technology', 'Second'),
('S0003', 'Information Technology', 'Second'),
```

```
('S0004', 'Business Management', 'Third'),
('S0005', 'Engineering', 'Fourth');


INSERT INTO Faculty_Member (FID, Title,
Department) VALUES
('F0001', 'Professor', 'Computer Science'),
('F0002', 'Associate Professor', 'Biology'),
('F0003', 'Assistant Professor', 'Mathematics'),
('F0004', 'Lecturer', 'Physics'),
('F0005', 'Researcher', 'Chemistry');

INSERT INTO Staff (StaffID, Position, Office)
VALUES
('Sf001', 'Clerk', 'Main Office'),
('Sf002', 'Administrator', 'HR Office'),
('Sf003', 'System Administrator', 'IT Office'),
('Sf004', 'Supervisor', 'Maintenance'),
('Sf005', 'Receptionist', 'Front Desk');



INSERT INTO Librarian (LID, Department,
Designation) VALUES
('L0001', 'Engineering', 'Assistant'),
('L0002', 'Information Technology', 'Assistant'),
('L0003', 'Business Management', 'Assistant'),
('L0004', 'Information Technology', 'Manager'),
('L0005', 'Engineering', 'Manager');
```

```
INSERT INTO Item_Copy (Copy_ID, Condition, Shelf,
Item_ID) VALUES
('C0001', 'Good', 'Shelf A1', 'B0001'),
('C0002', 'Excellent', 'Shelf A2', 'B0002'),
('C0003', 'Fair', 'Shelf B1', 'B0003'),
('C0004', 'New', 'Shelf B2', 'B0004'),
('C0005', 'Good', 'Shelf C1', 'B0005'),
('C0006', 'Fair', 'Shelf C2', 'J0001'),
('C0007', 'Excellent', 'Shelf D1', 'J0002'),
('C0008', 'New', 'Shelf D2', 'J0003'),
('C0009', 'Good', 'Shelf E1', 'J0004'),
('C0010', 'Excellent', 'Shelf E2', 'J0005'),
('C0011', 'Fair', 'Shelf F1', 'M0001'),
('C0012', 'New', 'Shelf F2', 'M0002'),
('C0013', 'Good', 'Shelf G1', 'M0003'),
('C0014', 'Excellent', 'Shelf G2', 'M0004'),
('C0015', 'Fair', 'Shelf H1', 'M0005');


INSERT INTO Library_Branch (Branch_ID,
Branch_Name, Branch_Location) VALUES
('B001', 'Central Library', 'New building Level
1'),
('B002', 'East Branch', 'Main building Level 3'),
('B003', 'West Branch', 'BM building Level 4'),
('B004', 'North Branch', 'New building Level 2'),
('B005', 'South Branch', 'New building Level 3');
```

```
INSERT INTO Library_Has_Copy (Copy_ID, Branch_ID)
VALUES
('C0001', 'B001'),
('C0002', 'B002'),
('C0003', 'B003'),
('C0004', 'B004'),
('C0005', 'B005');
```

```
INSERT INTO Category (Category_ID, Category_Name)
VALUES
('Cat01', 'Fantasy'),
('Cat02', 'Science fiction'),
('Cat03', 'Science'),
('Cat04', 'Comedy'),
('Cat05', 'Romantic');
```

```
INSERT INTO Item_Belongs_Category (Item_ID,
Category_ID) VALUES
('B0001', 'Cat01'),
('B0002', 'Cat02'),
('B0003', 'Cat03'),
('B0004', 'Cat04'),
('B0005', 'Cat05'),
```

```
('J0001', 'Cat01'),
('J0002', 'Cat02'),
('J0003', 'Cat03'),
('J0004', 'Cat04'),
('J0005', 'Cat05'),
('M0001', 'Cat01'),
('M0002', 'Cat02'),
('M0003', 'Cat03'),
('M0004', 'Cat04'),
('M0005', 'Cat05');
```

```
INSERT INTO Review (UserID, Item_ID, Review_Date,
Rating, Comment) VALUES
('S0001', 'B0001', '2024-10-10', '5', 'Excellent
read'),
('S0002', 'B0002', '2024-10-11', '4', 'Very
informative'),
('S0003', 'B0003', '2024-10-12', '3', 'Good
content'),
('S0004', 'B0004', '2024-10-13', '5', 'Loved
it'),
('S0005', 'B0005', '2024-10-14', '2', 'Not my
favorite');
```

```
INSERT INTO Msg (MID, Msg, LID) VALUES
('00001', 'Library is closing soon.', 'L0004'),
('00002', 'New books arrived.', 'L0004'),
```

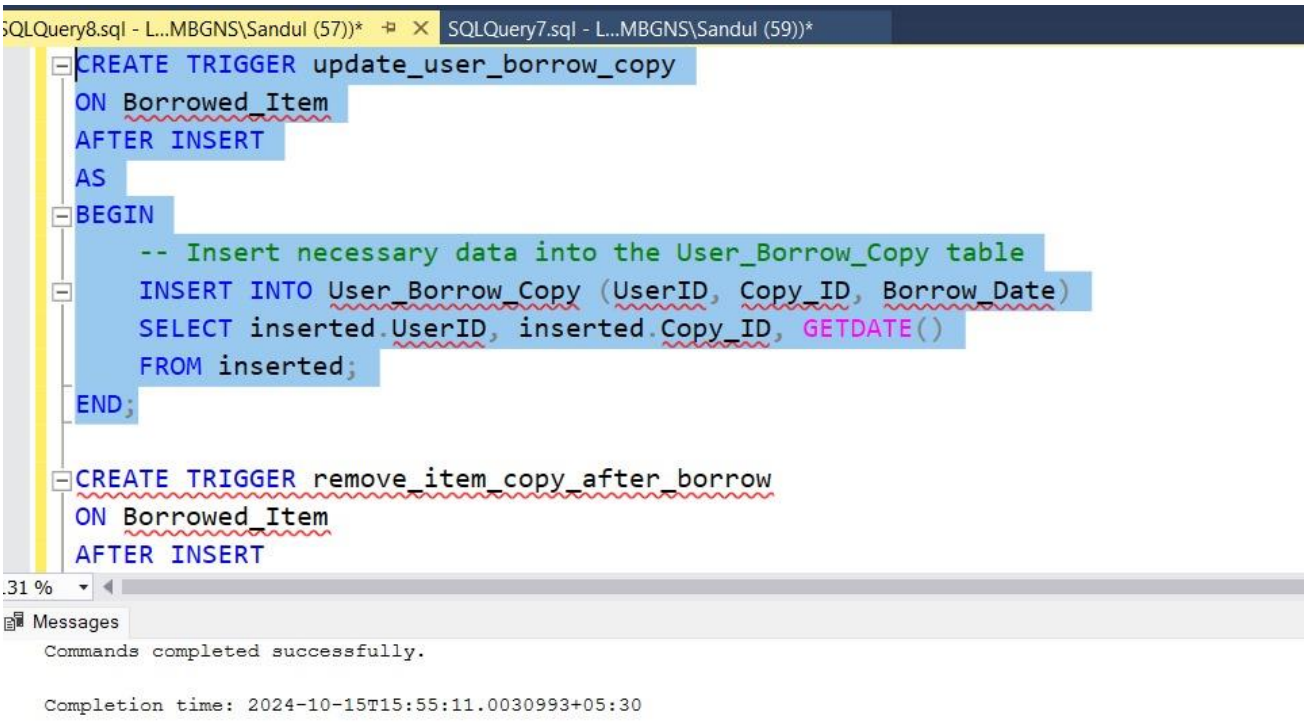
```
('00003', 'Staff meeting at 10 AM.', 'L0005'),
('00004', 'Library event next week.', 'L0005'),
('00005', 'Holiday hours posted.', 'L0005');


INSERT INTO Msg_Deliver_LUser (UserID, MID)
VALUES
('S0001', '00001'),
('S0002', '00002'),
('S0003', '00003'),
('S0004', '00004'),
('S0005', '00005');


INSERT INTO Borrowed_Item (Borrow_ID, Due_Date,
Borrow_Status, Copy_ID, UserID) VALUES
('Br001', '2024-10-01', 'Borrowed', 'C0001',
'S0001'),
('Br002', '2024-10-02', 'Borrowed', 'C0002',
'S0002'),
('Br003', '2024-10-03', 'Returned', 'C0003',
'S0003'),
('Br004', '2024-10-04', 'Borrowed', 'C0004',
'S0004'),
('Br005', '2024-10-05', 'Returned', 'C0005',
'S0005');
```

```sql
INSERT INTO Returned (Return_ID,
Return_Condition, Return_Date, Borrow_ID)
VALUES
('R0001', 'Good', '2024-10-02', 'Br001'),
('R0002', 'Fair', '2024-10-02', 'Br002'),
('R0003', 'Excellent', '2024-10-02', 'Br003'),
('R0004', 'Good', '2024-10-02', 'Br004'),
('R0005', 'Damaged', '2024-10-02', 'Br005');

INSERT INTO Fine (Fine_ID, Amount, Reason,
Return_ID)
VALUES
('F0001', 50.00, 'Late return', 'R0001'),
('F0002', 90.00, 'Damaged', 'R0005');


INSERT INTO User_Borrow_Copy (UserID, Copy_ID,
Borrow_Date)
VALUES
('S0001', 'C0001', '2024-01-15'),
('S0002', 'C0002', '2024-01-20'),
('S0003', 'C0003', '2024-01-22'),
('S0004', 'C0004', '2024-01-25'),
('S0005', 'C0005', '2024-01-30');
```

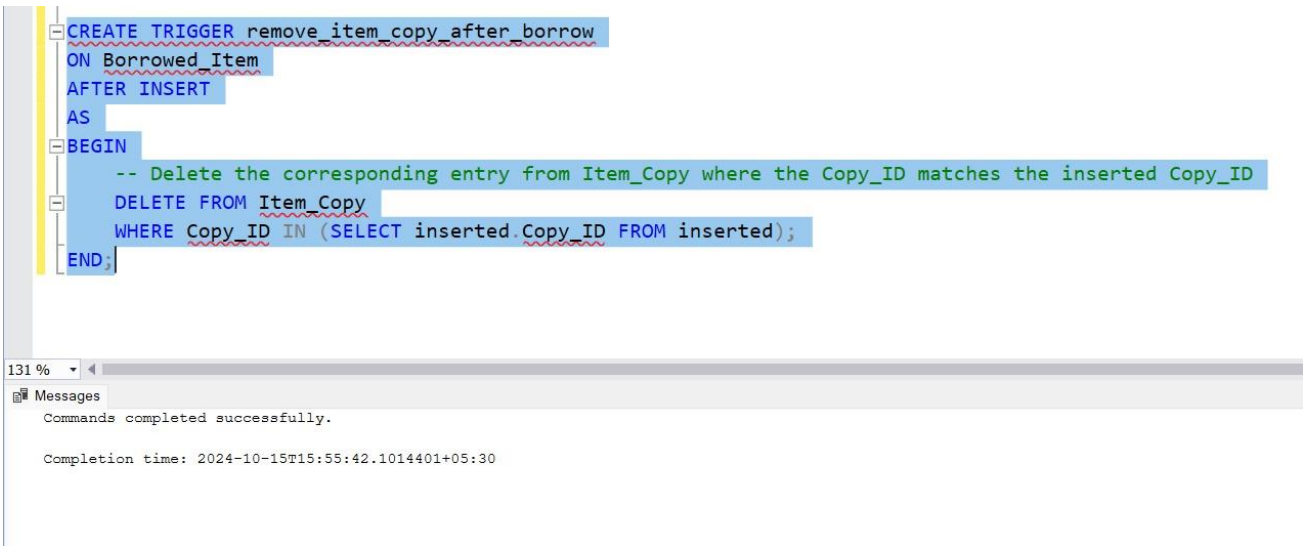
7.3. Database Programming

7.3.1. Trigger

This trigger will ensure that once a record is added to the Borrowed_Item table, it inserts the relevant data (UserID, Copy_ID, and Borrow_Date) into the User_Borrow_Copy table. The Borrow_Date will be the current date, as requested.



This trigger will fire after an insert into the Borrowed_Item table and then delete the corresponding entry from the Item_Copy table based on the Copy_ID



7.3.2. View

SQLQuery9.sql - L...MBGNS\Sandul (52))* | SQLQuery8.sql - L...MBGNS\Sandul (57))* | SQLQuery7.sql - L...MBG

```
CREATE VIEW Item_Category
AS
SELECT i.Item_Name, c.Category_Name
FROM Item i, Category c, Item_Belongs_Category b
WHERE i.Item_ID = b.Item_ID and b.Category_ID = c.Category_ID
```

131 %

Messages

```
Commands completed successfully.

Completion time: 2024-10-15T16:04:33.2580649+05:30
```

SQLQuery10.sql -...MBGNS\Sandul (63))* | SQLQuery9.sql - L...MBGNS\Sandul (52))* | SQLQuery8.sql - L...MBGNS\Sandul (57))* | SQLQuery7.sql -

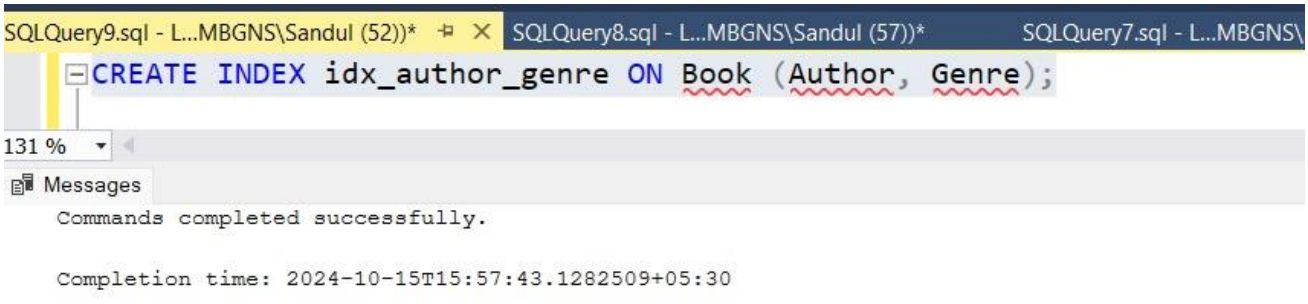
```
CREATE VIEW Book_Shelf
AS
SELECT i.Item_Name, l.Branch_Name, c.Shelf
FROM Item i, Library_Branch l, Item_Copy c, Library_Has_Copy h
WHERE i.Item_ID = c.Item_ID and c.Copy_ID = h.Copy_ID and h.Branch_ID = l.Branch_ID
```

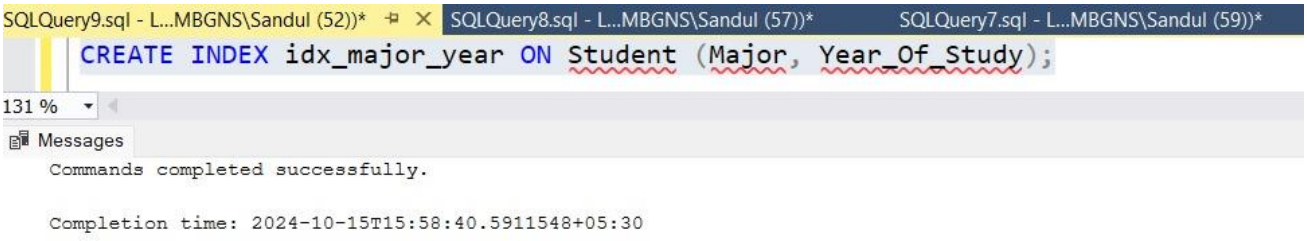
131 %

Messages

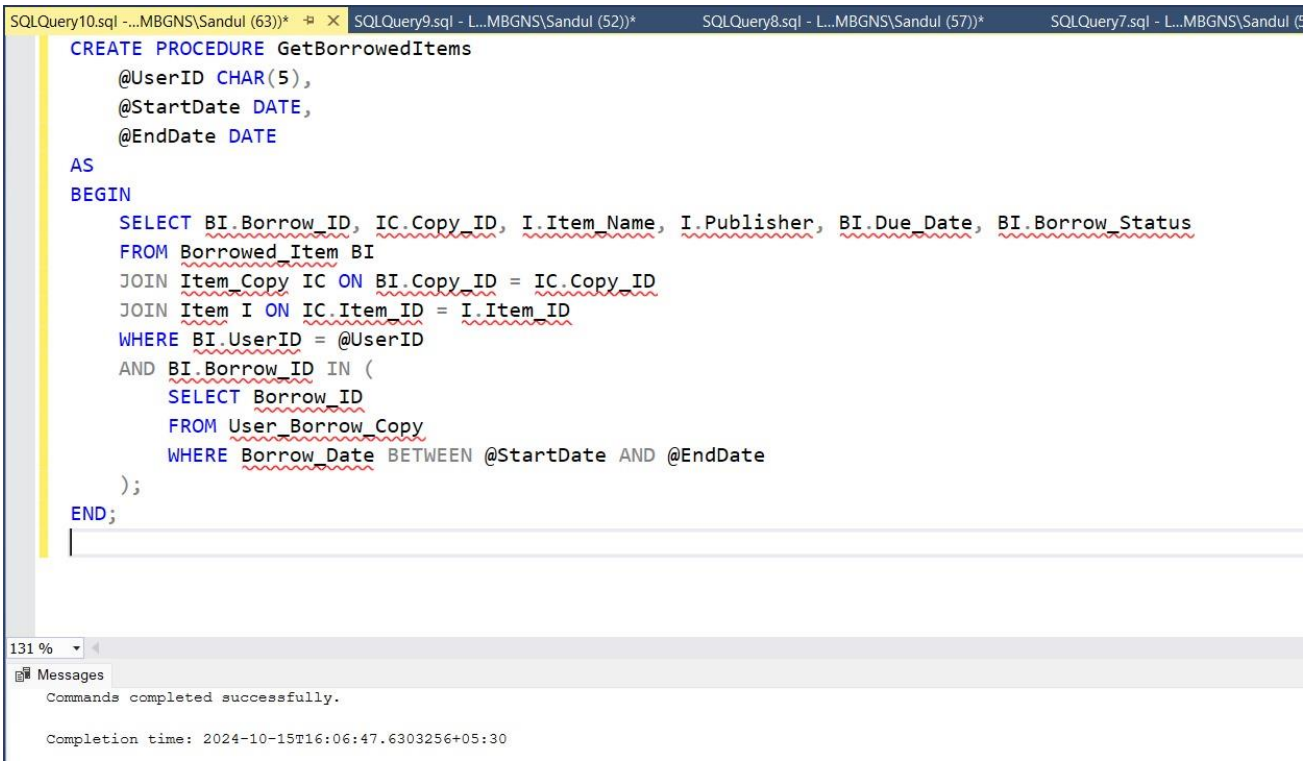
```
Commands completed successfully.

Completion time: 2024-10-15T16:07:52.2747483+05:30
```

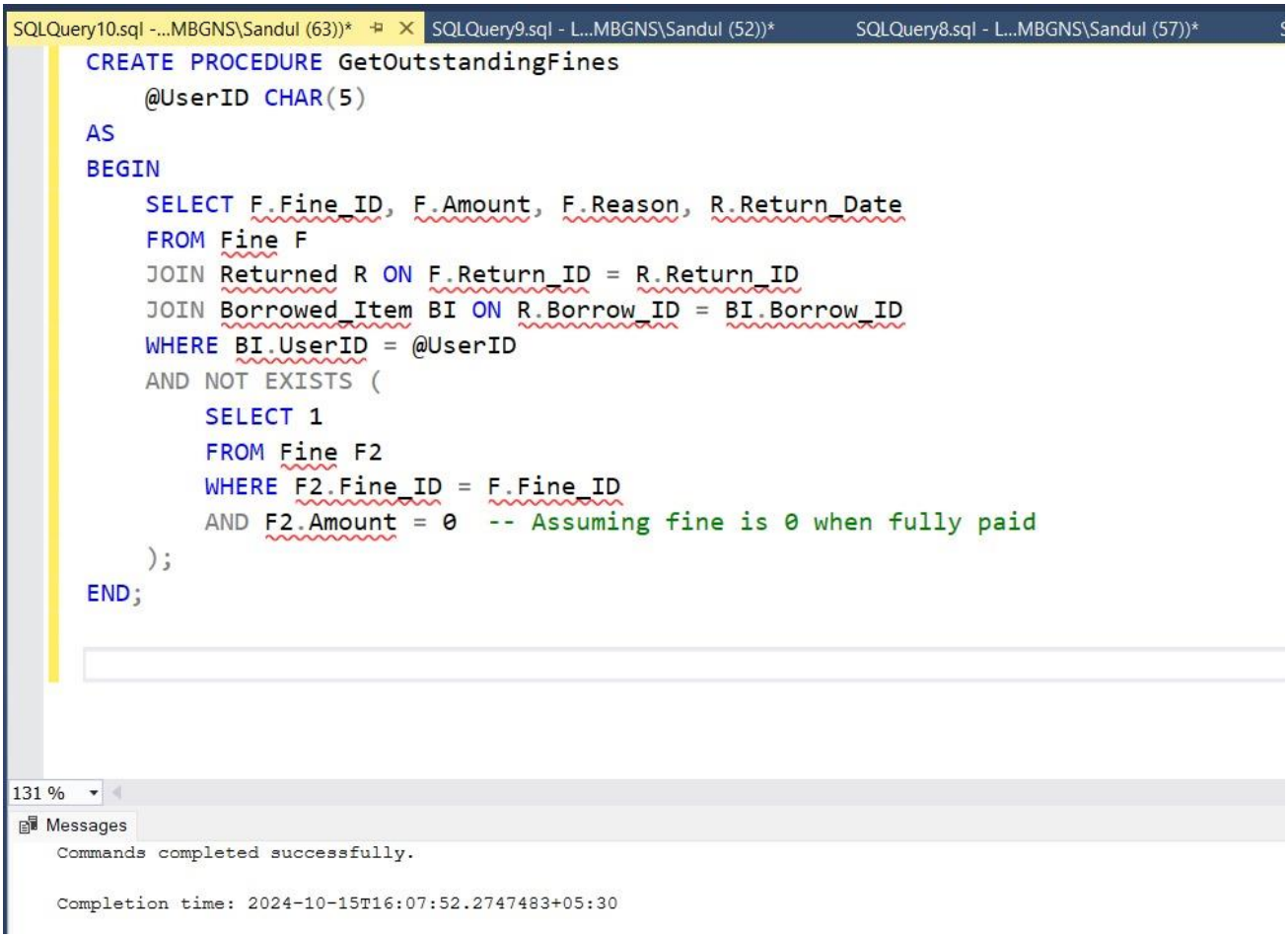
7.3.3. Index


SQLQuery9.sql - L...MBGNS\Sandul (52))*
SQLQuery8.sql - L...MBGNS\Sandul (57))*
SQLQuery7.sql - L...MBGNS\
CREATE INDEX idx_author_genre ON Book (Author, Genre);
131 %
Messages
Commands completed successfully.
Completion time: 2024-10-15T15:57:43.1282509+05:30


SQLQuery9.sql - L...MBGNS\Sandul (52))*
SQLQuery8.sql - L...MBGNS\Sandul (57))*
SQLQuery7.sql - L...MBGNS\Sandul (59))*
CREATE INDEX idx_major_year ON Student (Major, Year_Of_Study);
131 %
Messages
Commands completed successfully.
Completion time: 2024-10-15T15:58:40.5911548+05:30

7.3.4. Procedures





8. Database Vulnerabilities

8.1. SQL INJECTION

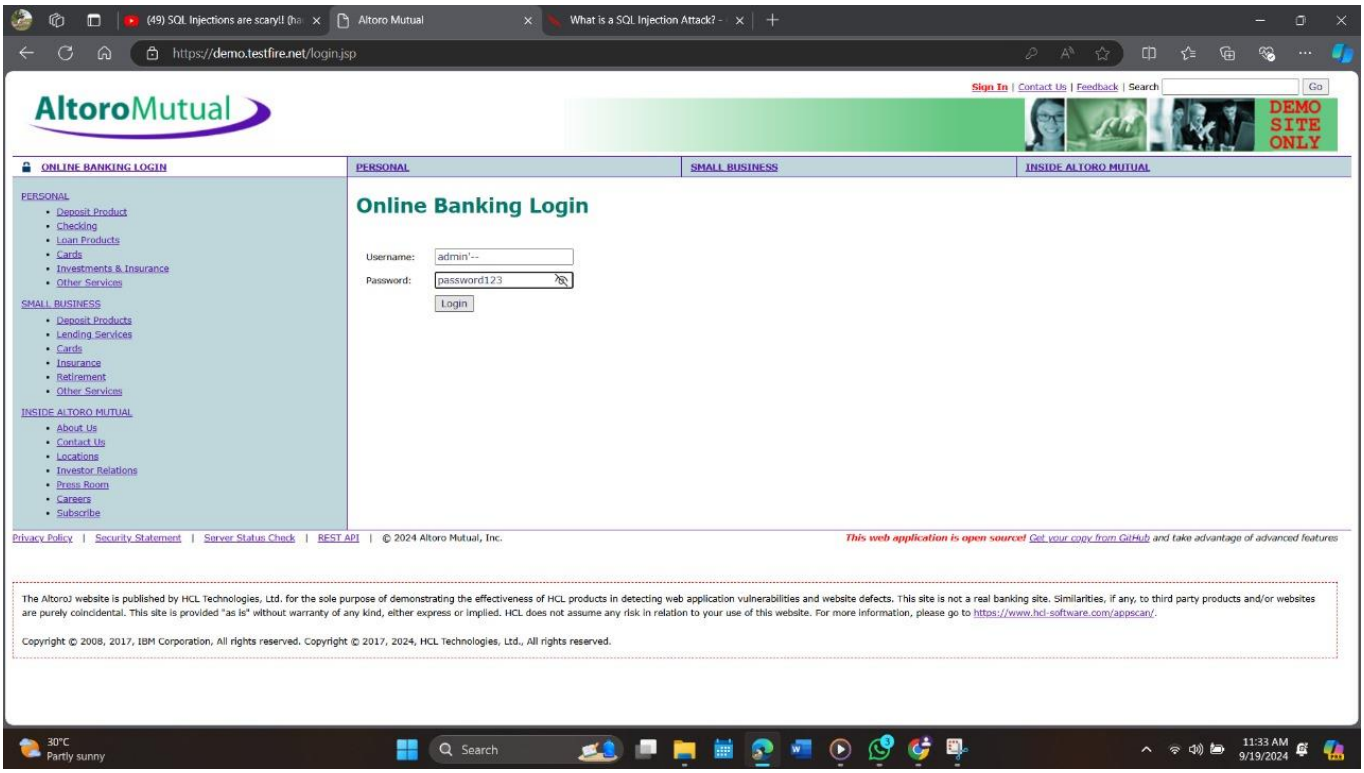
8.1.1. Introduction and Practical Example

SQL injection is **an information security attack that injects malicious SQL code into an application, it will affect confidentiality, integrity, availability (CIA Triad).** According to data in 2021, there were 274000 SQL injection happened and sql injection attack are 3rd most serious web application security risk in that year.

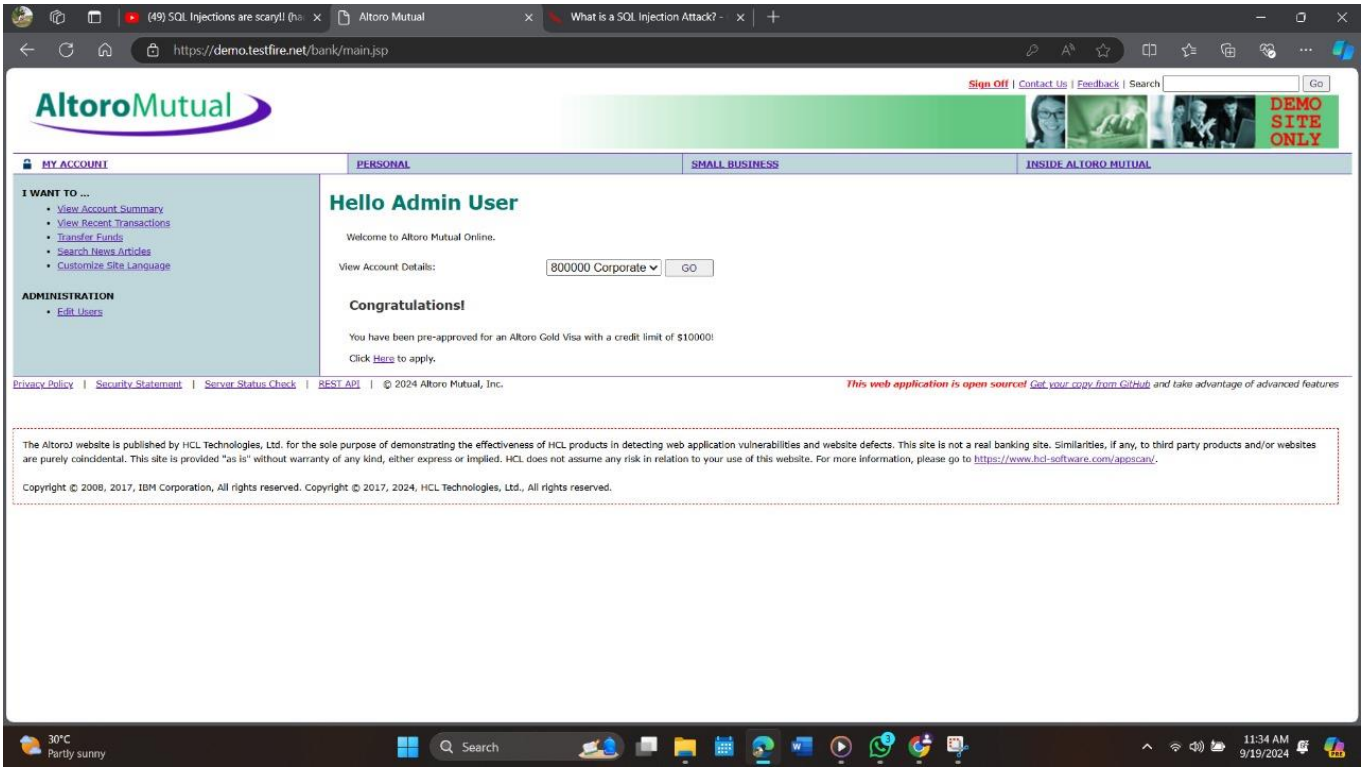
Practical example 1:

To do a demo about SQL injection we need a target site for learning and testing, so this site will help us to legally hack into website. Altoro Mutual (testfire.net)

1. give username as: admin'--
2. give password as: password123 (for this can use any)



3.click logging. Then redirect to user profile.



Why this happened?

```
<?php
// Create connection
$conn = new mysqli("localhost", "username", "password", "database");

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Vulnerable query, directly inserting user input into SQL
$username = $_POST['username'];
$password = $_POST['password'];

$sql = "SELECT * FROM users WHERE username = '$username' AND password =
'$password'";
$result = $conn->query($sql);

// Check if a user was found
if ($result->num_rows > 0) {
   echo "Login successful!";
} else {
   echo "Invalid username or password.";
}

// Close the connection
$conn->close();
?>
```

Reason1:
Variables $password, $username are didn't use code validations.

Reason2:
admin' -- as the username will comment the password part, so without correct password we can log in as admin.

Select *

From user

Where username= 'admin' --' AND password= 'passwod123';

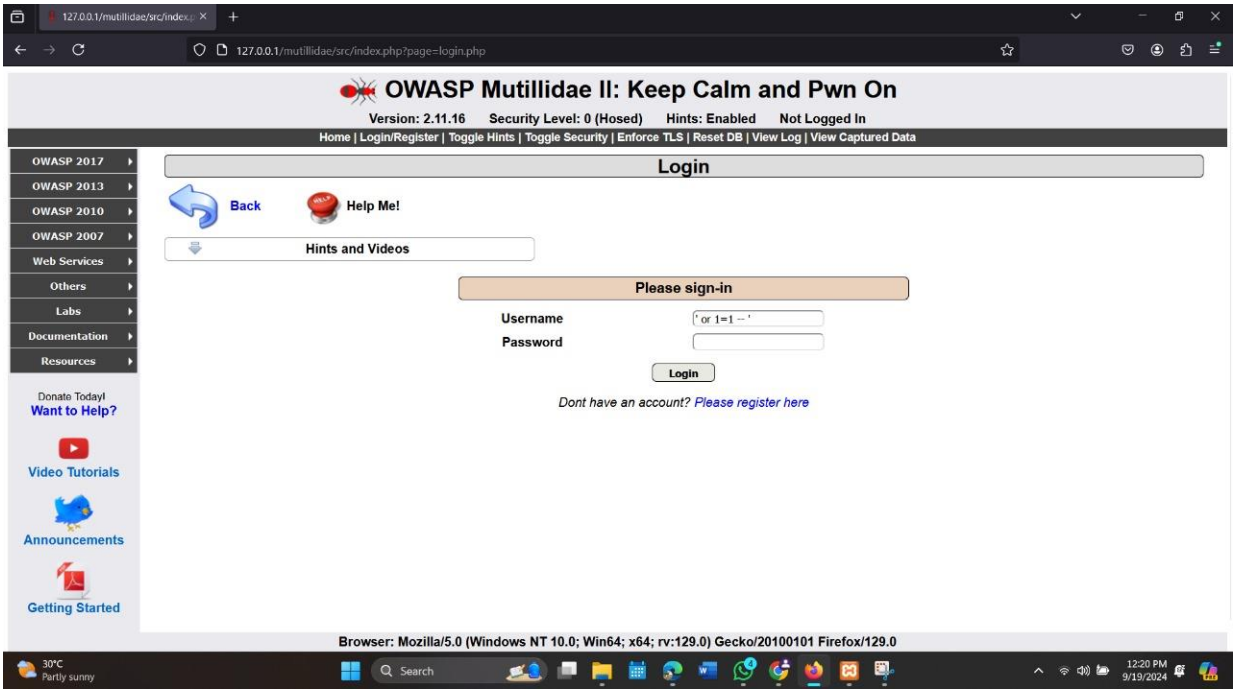
How to mitigate this:

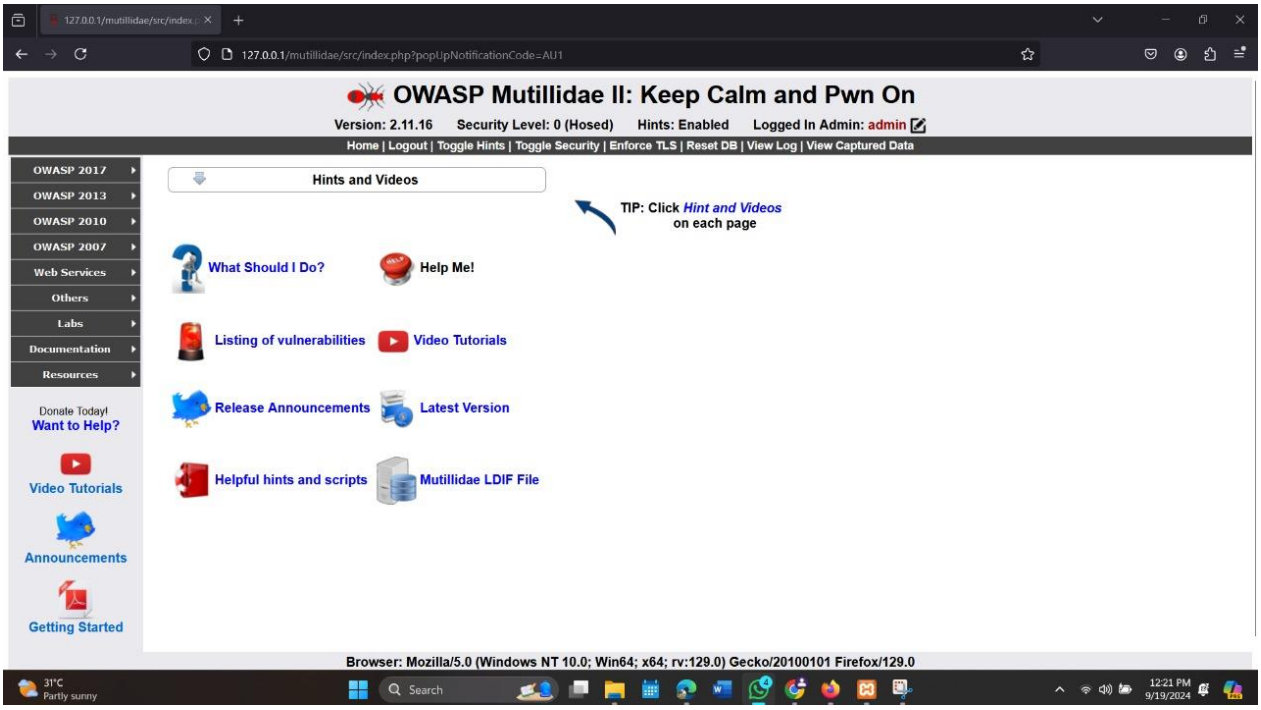
```
<?php
// Create connection
$conn = new mysqli("localhost", "username", "password", "database");

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Prepare the statement to prevent SQL injection
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password
= ?");
$stmt->bind_param("ss", $username, $password);

// Get the username and password from user input
$username = $_POST['username'];
$password = $_POST['password'];

// Execute the prepared statement
$stmt->execute();

// Get the result
$result = $stmt->get_result();

// Check if a user was found
if ($result->num_rows > 0) {
   echo "Login successful!";
} else {
   echo "Invalid username or password.";
}

// Close the statement and connection
$stmt->close();
$conn->close();
?>
```

Practical example 2:

To do a demo about SQL injection we need a target site for learning and testing, so this site will help us to legally hack into website. https://github.com/webpwnized/mutillidae.git

1. give username as: ‘ or 1=1 -- ’
2. give password as: blank
3.Rederected to user profile





Why this happen?

Condition or 1=1 always true, So it will bypass the login to attacker.

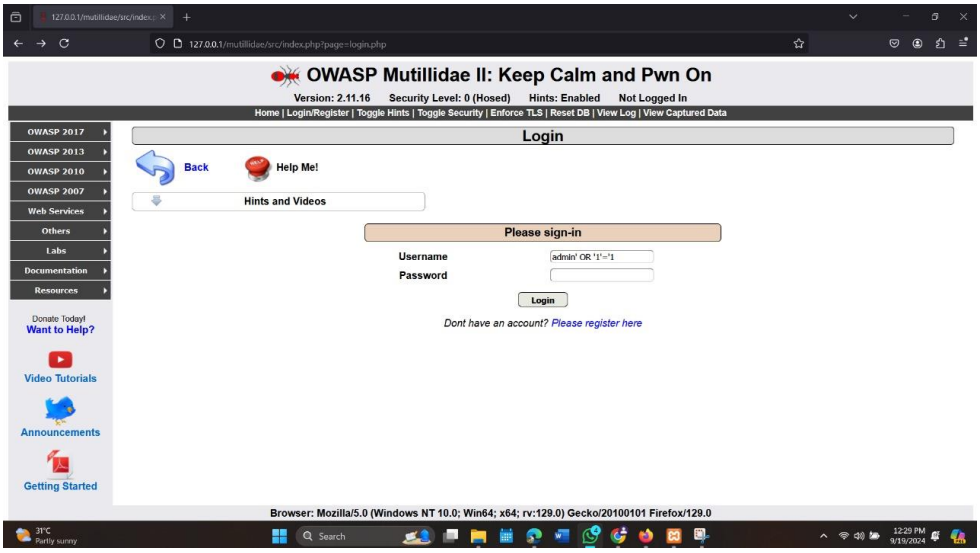
Username	' or 1=1 -- '

Select *

From users

Where username = ‘ ’ OR 1=1 --’ AND password = ‘ ’;

Practical example 3:



1. give username as: admin’ OR ‘1’=‘1
2. give password as: blank
3.Rederected to user profile.

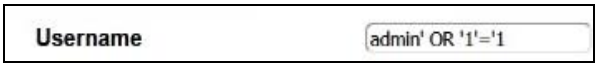
Why this happen?

- AND is always priority than OR, so ‘1’=’1’ AND password= ‘ ’ is checked first.

1=1 always true, and password empty is always true. So that means that part over roll true. Like:

Where username = ‘admin’ OR true;

- So that true with anything going via OR condition is TRUE. So, the attacker can bypass.



Select *

From users

Where username = ‘admin’ OR ‘1’=‘1’ AND password = ‘ ’;

8.1.2. Impact of Sql injetion Vulnerabilities

1. Data Theft
    - Exposure of Sensitive Data:

    There's a possibility that an attacker can fetch sensitive data like customer information, such as personal details, credit card numbers, passwords, among others, from the database. These all add to privacy breaches and potential regulatory financial penalties. For instance, this would be a violation under GDPR or HIPAA.

    - Intellectual Property Theft:

    This may involve the exposure of confidential business information in the form of trade secrets or proprietary information to your competitors or malicious entities.

2. Authentication Bypass
    - Login Bypass:

    Attackers utilize SQLi in bypassing security authentication mechanisms through manipulating forms used for logging in—for instance, logging in as an administrator without knowing the proper credentials.

    - Privilege Escalation:

    Poor or incorrect access controls allow attackers, upon their manipulation through SQLi, to gain privileges beyond what was intended.

3. Regulatory and Legal Implications
    - Non-compliance Fines:

    SQLi-related breaches can lead to violations of different data protection laws, such as GDPR, CCPA, or HIPAA, which will probably be coupled with heavy fines and other legal implications.

    - Lawsuits:

    Customers and partners can sue companies suffering from a data breach for not being able to protect sensitive information.

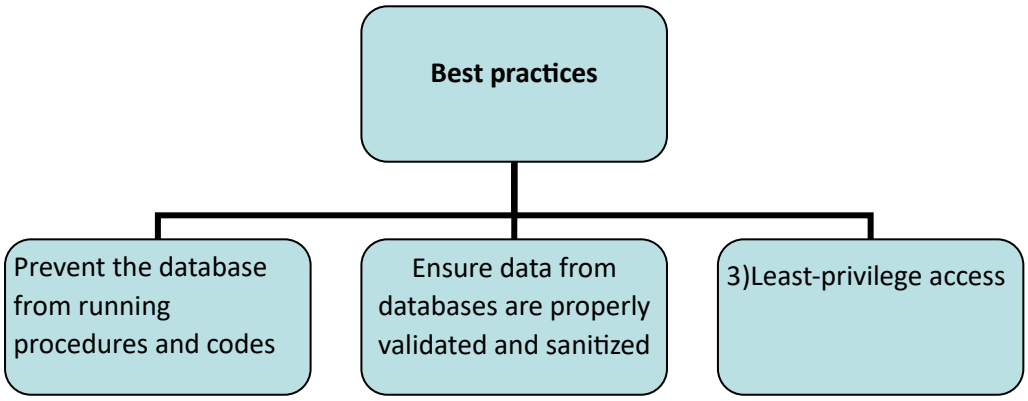
4. Reputation
    - Damage due to Loss of Customer Trust:

    A serious breach may lead to a loss of customer trust, which is usually very costly to restore and difficult.

    - Brand Damage:

Knowledge of a successfully exploited SQLi vulnerability can result in brand damage, with attendant loss of business and market value.

8.1.3. Best practices to protect database from SQL injection



1)Prevent the database from running procedures

a) Enforcing prepared statements and parameterized queries:

Prepared statements describe an acceptable SQL code, and then indicate specific parameters for received inquiries. Any and all SQL statements which are malicious are considered to be invalid data inputs, not executable commands.

2)Ensure data from databases are properly validated and sanitized

Any inputs that users feed into any SQL database should be checked, verified and free from any malicious codes as often as needed. Input validation is a regulation that is implemented to make sure that data undergo proper review and formatting based on pre-established rules of compliance Input sanitizing on the other hand, alters or 'cleans' the input to remove any improper or risky characters and to reshape the data as required. Ways of ensuring input validation include:

a) Establishing an allowlist:

An allowlist can help illustrate all the valid user inputs, by which the database can compare and verify the incoming queries that look strange. For example, special characters and extended URL are two types of inputs which are created by the user and can be dangerous for the collection of information from a particular database before executing other malicious statements. Restriction on the usage of such inputs may reduce the risks of an attack to the minimum.

3)Least-privilege access

is the ability of allowing users only the access level that is necessary for their work in the organization. For instance, this can be making sure that few users are permitted to have full access to data in a given

database and or allowing users to access data in a database on temporary basis, only to be pulled out later.

Limiting the ability of the users on an organizational role level also contributes to increase the complexity of a breach, as intruders who compromise a database via stolen login credentials will similarly be unable to read, write, steal or delete sensitive data. For the same reason, there is a necessity to restrict shared hosting of databases for multiple website and application.

8.2. Unpatched Vulnerabilities in Database Application

Technique:
Unpatched vulnerabilities occur when, there are some security flaws in a certain DBMS that have not been patched. These vulnerabilities arise due to codes errors or design flaws, which attackers can exploit to:

- Execute arbitrary code.
- Gain access to information without the proper authorization.
- They can – Provide a Denial of Service (DoS) attack.
- Privileges should be raised, and potentially even complete control over the DBMS may be achieved.

Real-World Scenario: Equifax Data Breach (2017)

Overview:
For instance, Equifax a credit reporting agency company which is among the largest was involved in one of the biggest data breaches in 2017. Highly sensational, as many as one hundred and forty-seven million plus people lost their credentials including Social Security numbers, birthdates, addresses etc. This breach occurred because Apache Struts, a web application framework that was in the use of Equifax had an un Patched Vulnerability. Although Equifax found a patch that was available two months before the attack, they did not apply it.

Impact:
1. Unauthorized Data Access: The attackers were able to transfer large of data potential identity thefts such as social security numbers, credit card numbers, the driver's license number among others.
2. Financial Loss: Due to the breach, Equifax incurred roughly $ 1.4 billion on fines regulatory, legal claims and enhancing measures for security.
3. Reputational Damage: As a result, the breach worsened the overall reputation and consumer perception of the company and Equifax.

8.2.1. Impact of Unpatched Vulnerabilities:

1. Data Breaches: The cyber criminals may penetrate through the systems and gains unauthorized access to the restricted documents and hence results to loss of customer information, financial records or any other property.

2. Denial of Service (DoS): impact to database accessibility(availability) this impacts highly critical applications and services.

3. Privilege Escalation: This means that attackers can fully exploit these weaknesses, which could lead to them having root or administrative access to the database environment, and manipulate, delete or have full control over the said environment.
4. System Compromise: When the system is not patched, other vulnerabilities like Remote Code Execution (RCE) enable the introduction of malicious software, further running of other undesirable scripts or even overall control of the base system.

8.2.2. Mitigation and Countermeasures

1. Patch Management

- Regular Updates: Visit the updates of database software and other parts associated with it such as Apache, OS, and libraries among others. Patch management can be done with patch management software that applies patches each time needed.
- Test Before Applying Patches: Before implementing security patches, it is recommended that one should run the implementation in a secluded environment to avoid any interferences or issues.

2. Automated Vulnerability Scanning:

- This means that is done automatically to identify weak points, vulnerable areas within the computer systems and applications.

- By using Qualys, Nessus or OpenVAS utility with the intention of assessing the database system for outstanding patches. These tools inform the administrators of missing patches or of an improperly secured system.

3. Database Hardening:

- Disable Unnecessary Features: Some of the recommendations made by the experts are some of the services, features and ports that should be disabled for instance, limit remote connection because it is not necessarily all the time. With these good practices, can reduce risk.

- Isolate the Database: Ensure that the database is not associated or recognizable through the other networks or the public internet. Use network segmentation to protect against intra network attack.

4. Intrusion Detection System and Intrusion Prevention System

- Use IDS/IPS solutions to monitor traffic and analyze them for possible attacks on the identified unpatched flaws. Such systems can detect attacks based on signatures and prevent exploitation in real-time processes.

5. Control of access and Role-Based Access Control (RBAC)

- Least Privilege: Minimize the privilege level of user accounts and implement the principle of using the minimum privilege level necessary to conduct day-to-day tasks. It must only allow those with the appropriate access levels to read or write to the database.

- Segmentation of Administrative Roles as admin and user account: This should help to reduce the risk exposure in case someone's account has been compromised.

6. Security Monitoring and Logging
   - Some of the database actions require log analysis, and it monitors unusual behaviors like logging failures, unexpected data browsing. In these scenarios, soc (security operation center) or IT Team can notify via their log analyzing tools and applications.

9. Conclusion:

The Equifax breach is a good example of what happens when organisations do not act to fix the weaknesses that are known to exist. Specifically, timely patching, regular vulnerability scanning, and database hardening are suggested to be applied to minimize the probabilities of an attack. It is therefore evident that organizations need to adhere to these mitigation measures and follow a strict security compliance so as to minimize the impact of unpatched database vulnerabilities that can compromise the integrity of data.

10. References:

What is a SQL Injection Attack? - CrowdStrike

How to prevent SQL injection | Cloudflare

11. Individual Contribution