

Library Database Business Requirements Document (BRD)

1. Introduction

The purpose of this document is to outline the business requirements for the development of a library management database system. This system aims to facilitate the library's operations, including book borrowing, genre and author management, staff responsibilities, and fine calculations for overdue books. The document specifies functional and non-functional requirements, assumptions, and constraints to ensure the system meets the library's needs effectively.

2. Business Goals

- To enable efficient borrowing and returning of books by members.
- To maintain accurate and up-to-date information about books, authors, genres, and loans.
- To ensure proper management of fines for overdue books.
- To enforce rules for membership registration and staff responsibilities.
- To support multi-author and multi-genre book classifications.

3. Functional Requirements

1. Membership Management:

- a. Members can borrow any number of books or none at all. Borrowing is optional and unrestricted in quantity.
- b. Members must provide a valid email or phone number to register.

2. Book and Genre Management:

- a. Each genre must have at least one associated book. Genres without books will be automatically removed from the database.
- b. Books can belong to multiple genres, and each genre can have any number of books.

3. Author Management:

- a. Each author must have at least one associated book. Authors without books will be automatically removed from the database.
- b. Books can be written by multiple authors, and each author can write multiple books.

4. Loan Management:

- a. A book can only be borrowed by one member at a time. Multiple members cannot borrow the same copy of a book simultaneously.
- b. Each loan is managed by one staff member.
- c. Members must return books within 14 days of the loan date.

5. Fine Management:

- a. Fines for overdue books are calculated at a fixed rate of \$12 per day starting from the 15th day.

6. Staff Management:

- a. Staff members can manage at least one loan and at most a defined maximum number of loans.
- b. Each staff member must have a hire date and a valid phone number.

4. Assumptions and Constraints

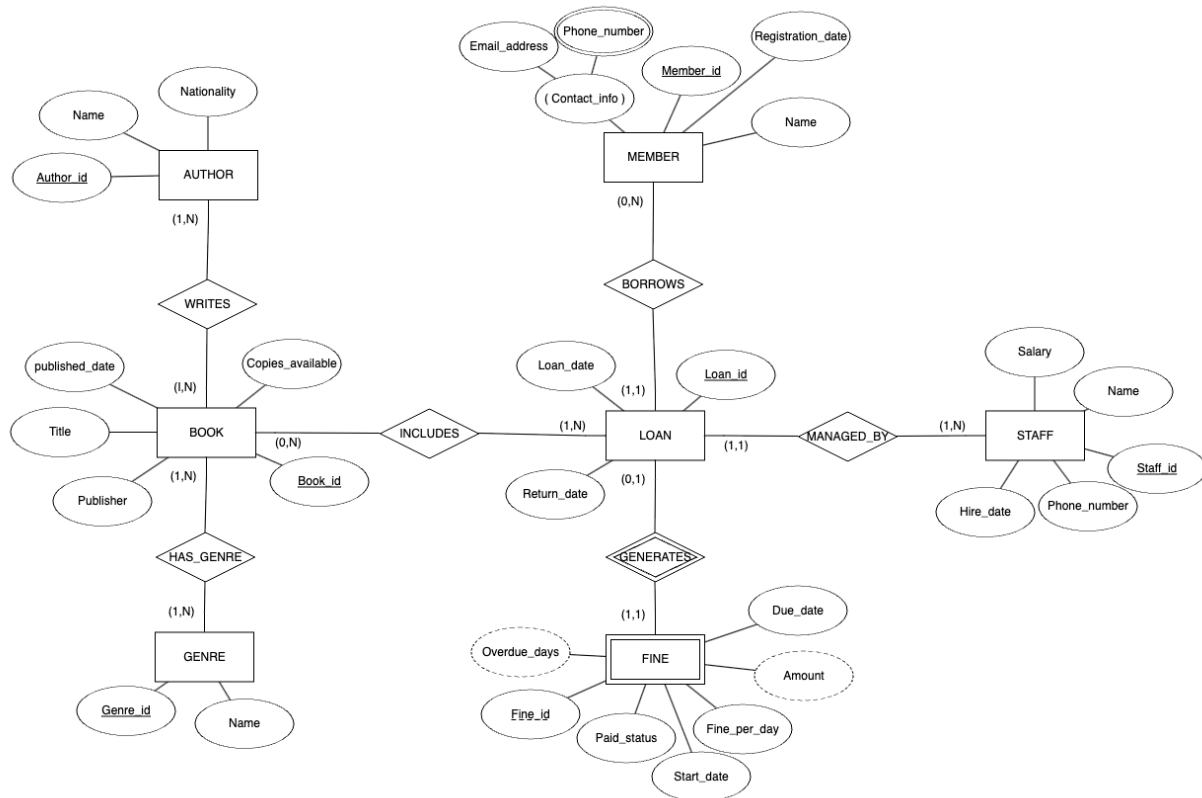
- Assumptions:
 - Members will return borrowed books in good condition.
 - Staff will only manage loans within their work hours.
 - Genres and authors will only be removed automatically when no books are associated with them.
- Constraints:
 - The library has a fixed fine rate of \$12 per day for overdue books.
 - Books must be returned within 14 days to avoid fines.
 - Each staff member has a predefined maximum number of loans they can manage at one time.

6. Notes

- **Book:** An item available for borrowing.
- **Member:** A registered user who can borrow books from the library.
- **Loan:** A record of a book borrowed by a member.
- **Genre:** A category or classification of books (Fiction, Non-Fiction).
- **Author:** The writer or contributor of a book.
- **Fine:** A monetary penalty for overdue books.

- **Staff Member:** An employee responsible for managing loans and other library operations.

ERD:

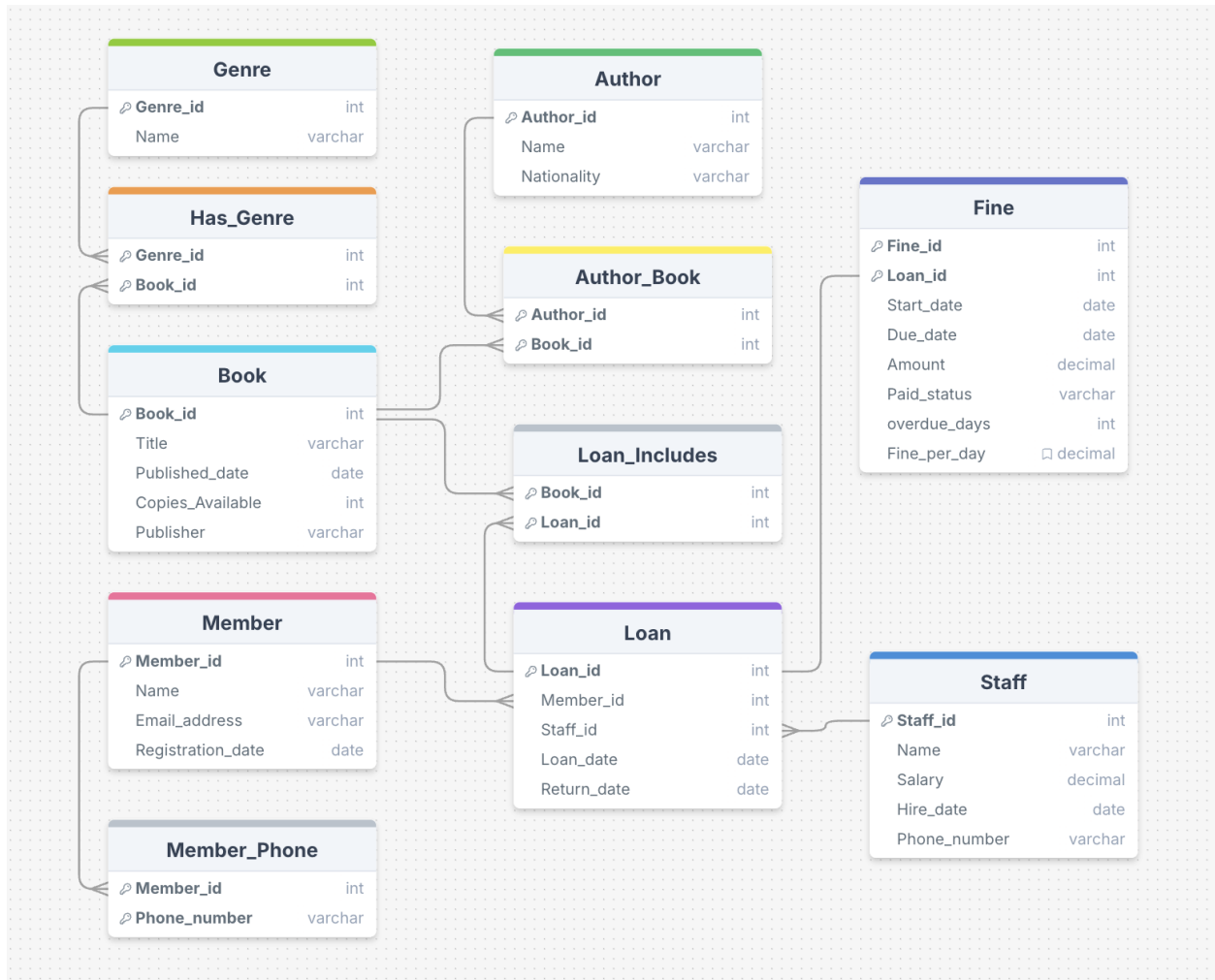


Database Design Document

Introduction:

The purpose of this document is to provide a detailed design for the Library Database system, which is intended to support the library's operational needs. The database will manage information about members, books, authors, genres, staff, loans, and fines. And will explain the relationship between them.

Schema:

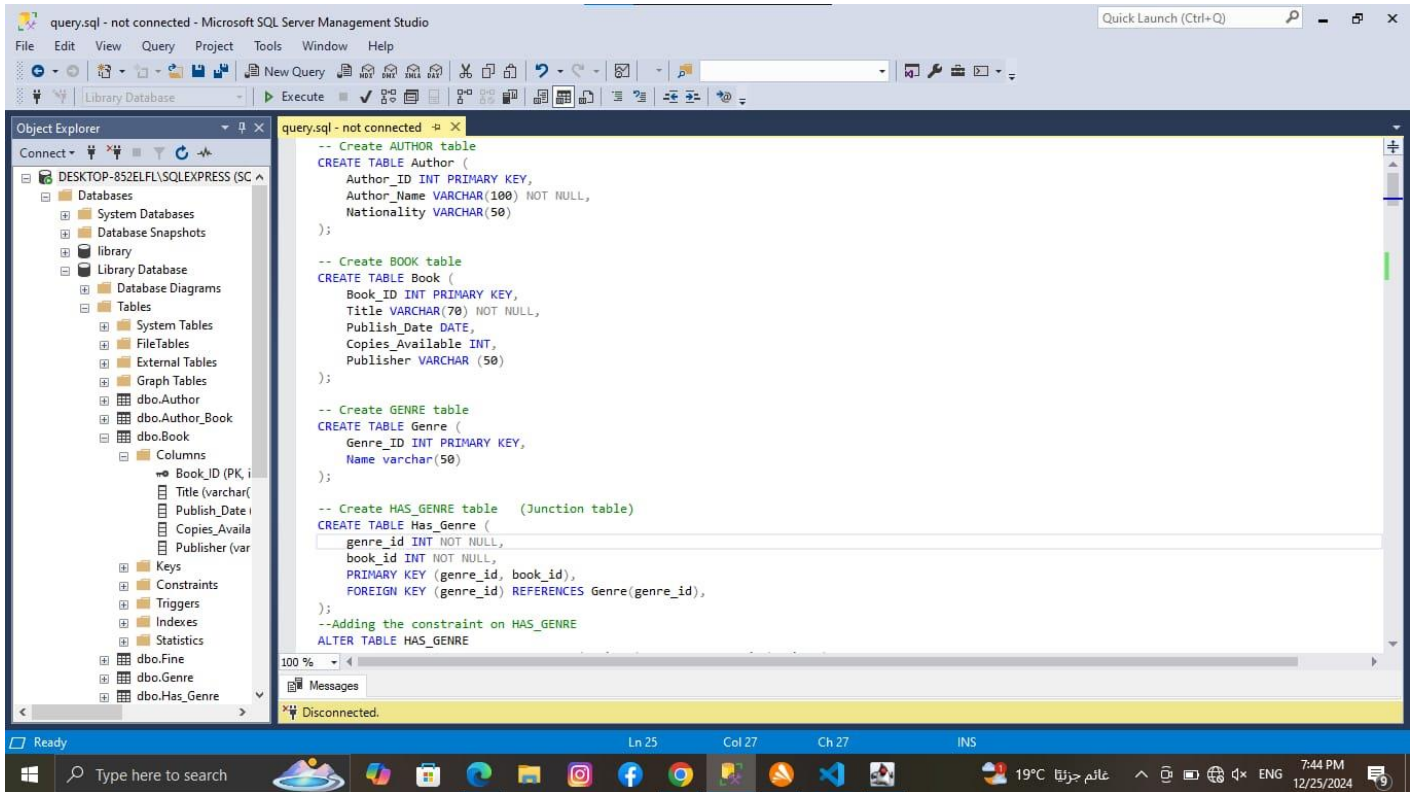


Assumptions:

1. The “phone number” is a multi-valued attribute. Therefore, a separated table is created to store “member id” and “phone number”, where “member id” and “phone number” is a composite primary key allowing a single member to have multiple phone numbers without worrying about duplications that might violate key constraints.
2. Since the relationship between the entity “**genre**” and the entity “**book**” is many-to-many, a junction table called “**has genre**” is created to hold the “book id” and “genre id” as foreign keys and they are also a composite primary key to manage the relationship between the two entities without violating key constraints.
3. Since the relationship between the entity “**book**” and the entity “**author**” is many-to-many, a junction table called “**Author_Book**” is created to hold the “book id” and “author id” as foreign keys and they are also a composite primary key to manage the relationship between the two entities without violating key constraints.
4. Since the relationship between the entity “**book**” and the entity “**loan**” is many-to-many, a junction table called “**Loan_Includes**” is created to hold the “book id” and “loan id” as foreign keys and they are also a composite primary key to manage the relationship between the two entities without violating key constraints.

5. A constraint called “ON DELETE CASCADE” was added to the foreign key in the “**Author_Book**” table, which is a junction table between the “**Book**” and “**Author**” tables. This ensures that when a book is deleted from the “**Book**” table, all corresponding records in the “**Author_Book**” table with the matching *Book_ID* will also be deleted, maintaining data integrity.
6. Similarly, the same “ON DELETE CASCADE” constraint was applied to the foreign key in the “**Has_Genre**” table, ensuring that if a book is deleted from the “**Book**” table, all corresponding records in the “**Has_Genre**” table with the matching “*Book_ID*” will also be deleted.

SQL Script



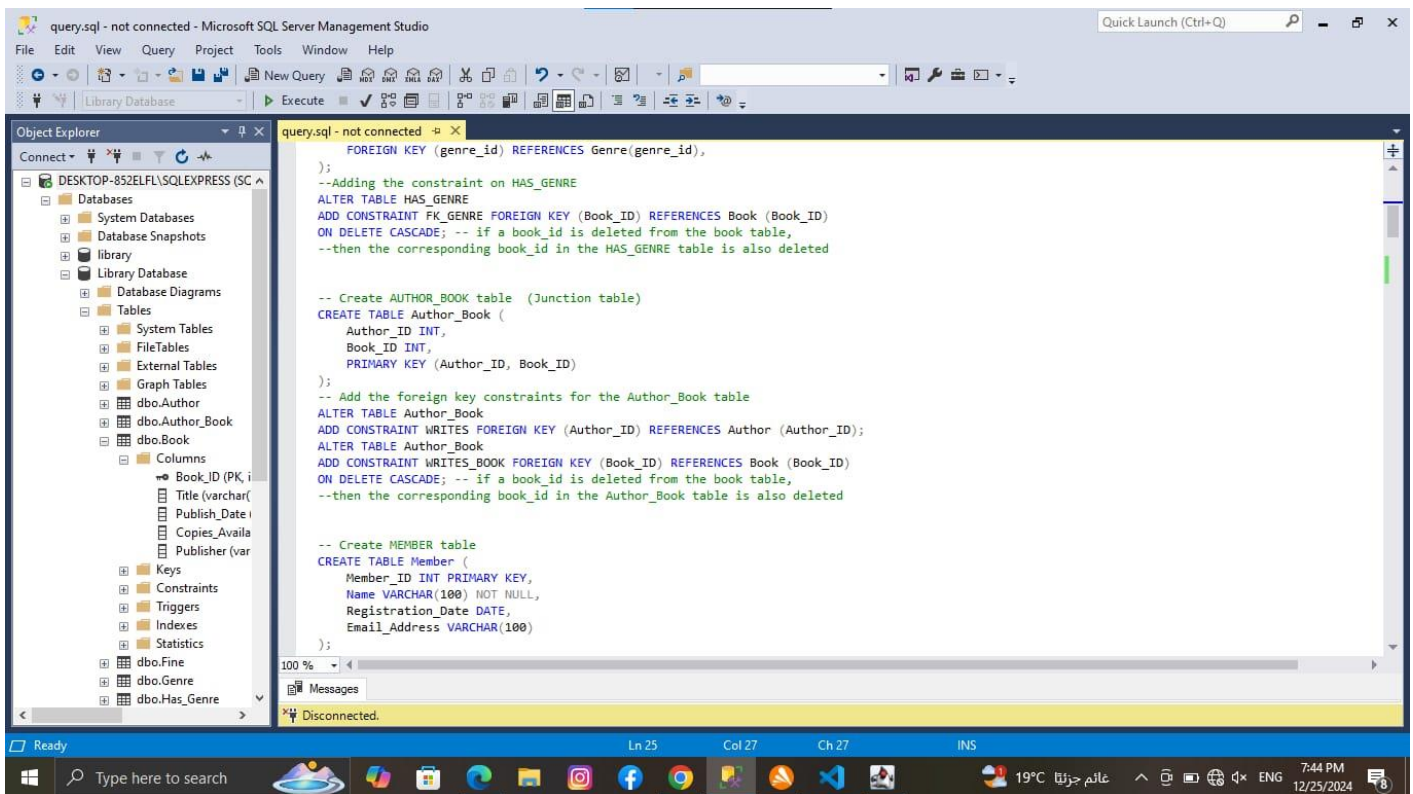
```
-- Create AUTHOR table
CREATE TABLE Author (
    Author_ID INT PRIMARY KEY,
    Author_Name VARCHAR(100) NOT NULL,
    Nationality VARCHAR(50)
);

-- Create BOOK table
CREATE TABLE Book (
    Book_ID INT PRIMARY KEY,
    Title VARCHAR(70) NOT NULL,
    Publish_Date DATE,
    Copies_Available INT,
    Publisher VARCHAR (50)
);

-- Create GENRE table
CREATE TABLE Genre (
    Genre_ID INT PRIMARY KEY,
    Name varchar(50)
);

-- Create HAS_GENRE table (Junction table)
CREATE TABLE Has_Genre (
    genre_id INT NOT NULL,
    book_id INT NOT NULL,
    PRIMARY KEY (genre_id, book_id),
    FOREIGN KEY (genre_id) REFERENCES Genre(genre_id),
);

--Adding the constraint on HAS_GENRE
ALTER TABLE HAS_GENRE
```



```
FOREIGN KEY (genre_id) REFERENCES Genre(genre_id),
);

--Adding the constraint on HAS_GENRE
ALTER TABLE HAS_GENRE
ADD CONSTRAINT FK_GENRE FOREIGN KEY (Book_ID) REFERENCES Book (Book_ID)
ON DELETE CASCADE; -- if a book_id is deleted from the book table,
--then the corresponding book_id in the HAS_GENRE table is also deleted

-- Create AUTHOR_BOOK table (Junction table)
CREATE TABLE Author_Book (
    Author_ID INT,
    Book_ID INT,
    PRIMARY KEY (Author_ID, Book_ID)
);

-- Add the foreign key constraints for the Author_Book table
ALTER TABLE Author_Book
ADD CONSTRAINT WRITES FOREIGN KEY (Author_ID) REFERENCES Author (Author_ID);
ALTER TABLE Author_Book
ADD CONSTRAINT WRITES_BOOK FOREIGN KEY (Book_ID) REFERENCES Book (Book_ID)
ON DELETE CASCADE; -- if a book_id is deleted from the book table,
--then the corresponding book_id in the Author_Book table is also deleted

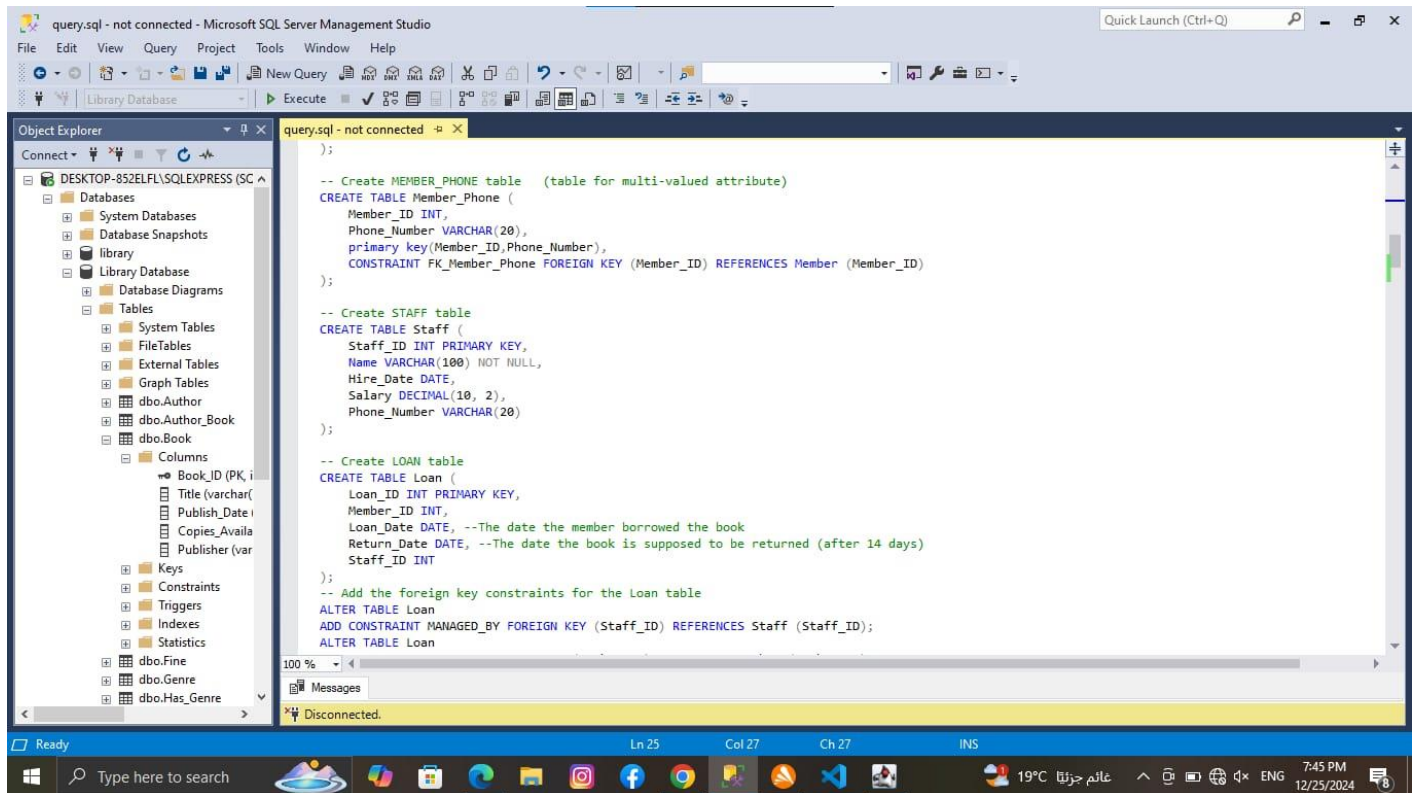
-- Create MEMBER table
CREATE TABLE Member (
    Member_ID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Registration_Date DATE,
    Email_Address VARCHAR(100)
);
```

We began by creating the necessary tables for the library database using the (CREATE TABLE) command. First, we defined the (Author) table with the

attributes (Author_ID, Author_Name, and Nationality), and set (Author_ID) as the primary key. Next, we followed a similar process to define the remaining tables.

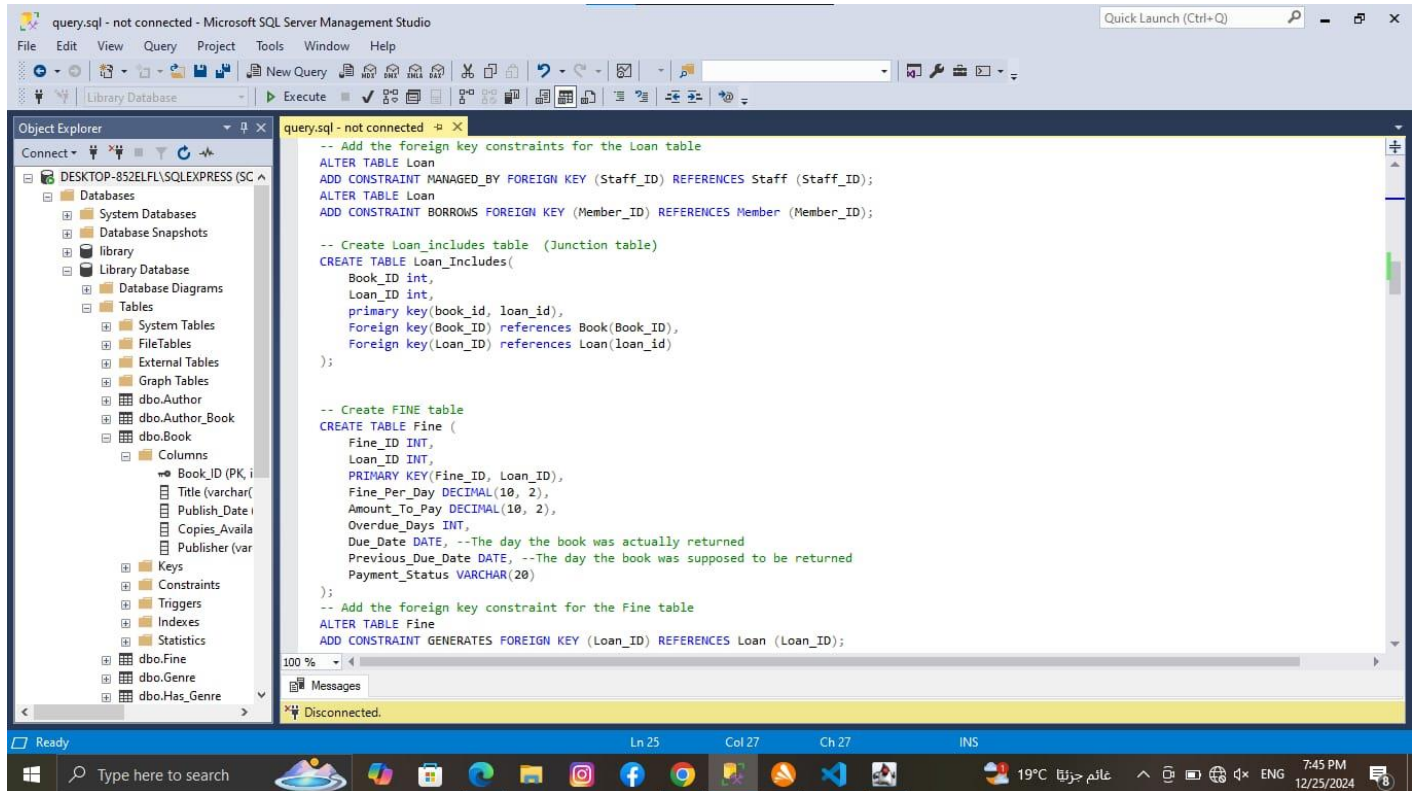
Since the relationship between the Genre and the Book table is many to many, we created the junction table (Has_Genre) to link the tables with a composite key of both their primary keys (genre_id) and (book_id) and we added a foreign key constraint to link the genre_id in the Has_Genre table to the genre_id in the Genre table using (FOREIGN KEY_REFERENCES) clause. Then, we added a foreign key constraint to the HAS_GENRE table to link the Book_ID attribute from the HAS_GENRE table to the Book_ID in the Book table, and we added the ON DELETE CASCADE here as well to ensure that if a book is deleted from the Book table, all corresponding entries in the Book_Genre table with that Book_ID will also be automatically deleted. Preventing having any genres that don't have any corresponding books.z

Since there is a many-to-many relationship between the Author and Book tables, we created the Author_Book table as a junction table. This table has a composite key consisting of the primary keys from both the Author and Book tables, to link authors to the books they have written. Then, we added foreign key constraints to the Author_Book table: 1. WRITES constraint: This links the Author_ID attribute from the Author_Book table to the Author_ID in the Author table to make sure that each record in the Author_Book table refers to a specific author. 2. WRITES_BOOK constraint: This links the Book_ID attribute from the Author_Book table to the Book_ID in the Book table. Then we used the ON DELETE CASCADE command so that if a book is deleted from the Book table, all corresponding records in the Author_Book table with that Book_ID will be automatically deleted. This helps maintain referential integrity by removing the relationship between authors and books when the book is removed.



Since the phone_number is a multivalued attribute in the Member table, we created the Member_Phone table to allow members to have multiple phone numbers associated with their accounts. We then set the PRIMARY KEY to be a composite key of both the Member_ID and the phone_number. We also added a foreign key constraint to reference the Member_ID from the Member table, establishing a relationship between the two tables.

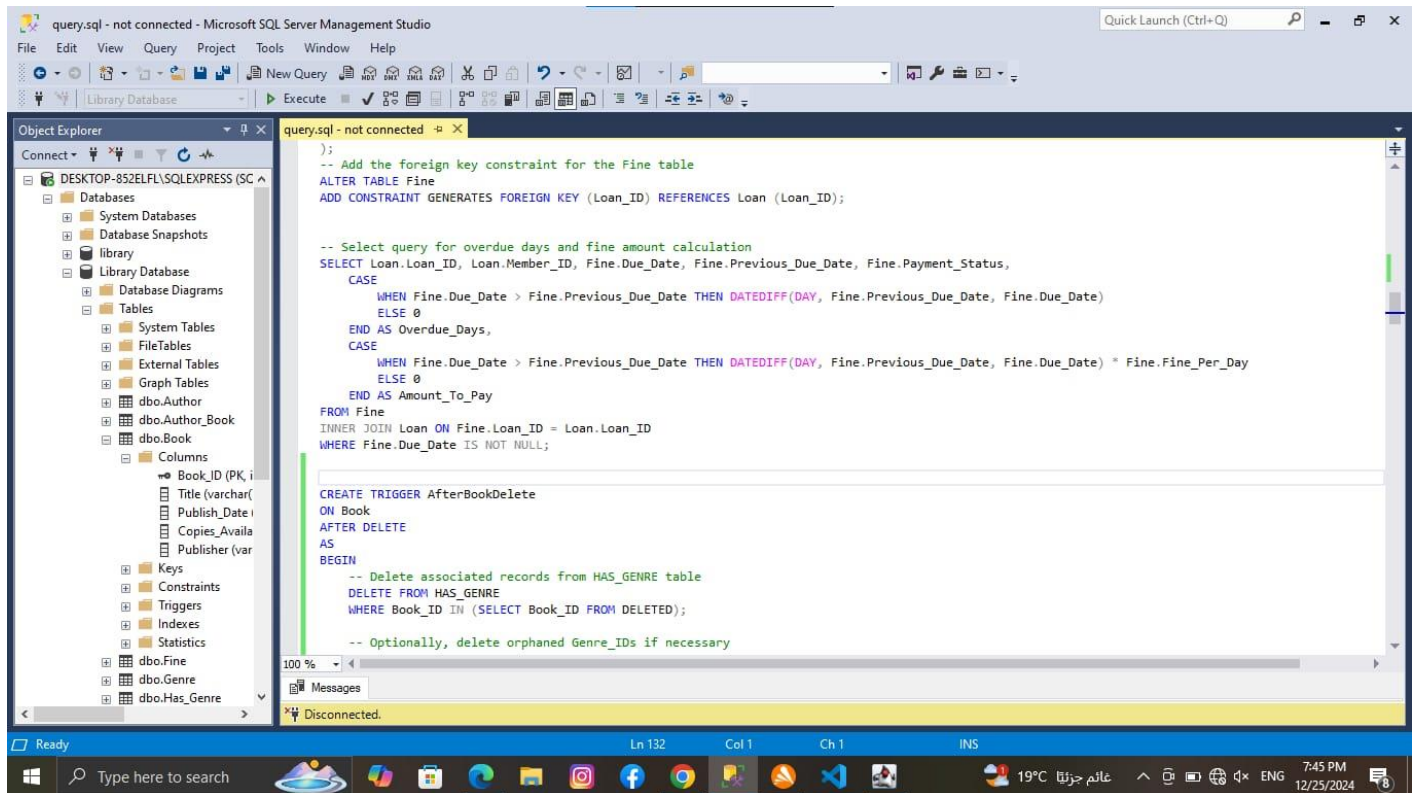
Here, we created the Loan table and set the PRIMARY KEY to be the Loan_ID, then we added the foreign-key constraints to the table to create those relationships: 1. MANAGED_BY constraint: This links the Staff_ID from the Staff table to the Staff_ID in the Loan table. It establishes that each loan is managed by a staff member. 2. BORROWS constraint: This links the Member_ID attribute from the Member table to the Member_ID in the Loan table to make sure that every loan is associated with a specific member.



We created the Loan_Includes table as a junction table to handle the many-to-many relationship between the Book and Loan tables. Which allows us to track which books are included in which loans.

Since the Fine table represents a weak entity, and its identifying entity is Loan, we created the Fine table with a composite primary key (Fine_ID, Loan_ID). Which makes sure that each Fine record is uniquely identified by the combination of both the Fine_ID and the Loan_ID.

Then we added the foreign key (GENERATES) constraint which links the Loan_ID attribute from the Loan table to the Loan_ID in the Fine table to ensure that each fine is associated with a specific loan.



Since the Overdue_Days and Amount_To_Pay attributes in the Fine table are derived, we calculated them using the following query. First, we retrieved the Loan_ID and Member_ID from the Loan table, along with the Due_Date, Previous_Due_Date, and Payment_Status from the Fine table using a SELECT statement. We then used a CASE-WHEN clause to apply a condition: if the Due_Date (the actual date the book was returned) is later than the Previous_Due_Date (the expected return date, which is 14 days after the loan date), it indicates that the book was returned late. In this case, we calculate the Overdue_Days by using the DATEDIFF function, which subtracts the Previous_Due_Date from the Due_Date to determine the number of overdue days. This value is stored in the Overdue_Days attribute. Finally, we calculate the Amount_To_Pay by multiplying the Fine_Per_Day by the Overdue_Days, determining the total fine amount for the member to pay. Then we used the INNER JOIN to combine the Fine table with the Loan table using the Loan_ID attribute as the common link. We ensured that only records with matching Loan_ID values in both tables were included in the results, as INNER JOIN only includes rows that have corresponding matches in both tables. Then we added a WHERE condition to make sure we only

include fines where the Due_Date is not null, meaning the book has actually been returned.

And this is the output of the query:

	Loan_ID	Member_ID	Due_Date	Previous_Due_Date	Payment_Status	Overdue_Days	Amount_To_Pay
1	1	1	2024-01-21	2024-01-15	Paid	6	72.00
2	2	2	2024-01-18	2024-01-17	Pending	1	12.00
3	3	3	2024-01-22	2024-01-19	Paid	3	36.00
4	8	8	2024-02-05	2024-02-01	Pending	4	48.00
5	10	10	2024-01-25	2024-01-23	Paid	2	24.00
6	11	11	2024-01-27	2024-01-26	Pending	1	12.00
7	12	12	2024-01-31	2024-01-29	Paid	2	24.00
8	15	15	2024-02-06	2024-02-04	Pending	2	24.00
9	16	16	2024-02-11	2024-02-08	Paid	3	36.00
10	17	17	2024-02-13	2024-02-10	Pending	3	36.00

```
-- Optionally, delete orphaned Genre_IDs if necessary
DELETE FROM Genre
WHERE Genre_ID NOT IN (
    SELECT DISTINCT Genre_ID
    FROM HAS_GENRE
);
END;
GO

CREATE TRIGGER AfterBookDelete2
ON Book
AFTER DELETE
AS
BEGIN
    -- Delete associated records from HAS_GENRE table
    DELETE FROM Author_Book
    WHERE Author_ID IN (SELECT Book_ID FROM DELETED);

    -- Optionally, delete orphaned Genre_IDs if necessary
    DELETE FROM Author
    WHERE Author_ID NOT IN (
        SELECT DISTINCT Author_ID
        FROM Author_Book
    );
END;
GO

-- Book
insert into Book (Book_ID, title, Publish_Date, publisher, Copies_available)
VALUES
(1, 'The Great Gatsby', '1925-04-10', 'Charles Scribners Sons', 5),
```

we created a trigger named AfterBookDelete. This trigger ensures that the database maintains integrity when a record is deleted from the Book table.

The trigger is activated immediately after a DELETE operation occurs on the Book table.

It automatically removes related records from the HAS_GENRE table to prevent invalid references (enforce referential integrity) and clean up the Genre table by deleting any genres that are no longer associated with any books.

When a book is deleted from the Book table, its Book_ID is temporarily stored in the virtual DELETED table. The DELETE statement removes all records from the HAS_GENRE table where the Book_ID matches any Book_ID in the DELETED table.

This ensures that the HAS_GENRE table does not contain references to books that no longer exist.

This second step ensures the Genre table does not contain "orphaned" genres (genres not associated with any books). The DELETE statement removes all genres from the Genre table where the Genre_ID is not found in the HAS_GENRE table.

We created another trigger named AfterBookDelete2, which is designed to maintain the integrity of the relationships between the Book, Author_Book, and Author tables in a database. This trigger is executed automatically after a record is deleted from the Book table.

This trigger is activated immediately after a DELETE operation occurs on the Book table. It automatically removes related records from the Author_Book table when a book is deleted and cleans up the Author table by deleting any authors who are no longer associated with any books.

The DELETED virtual table temporarily holds the Book_ID values of the books that were deleted.

The DELETE statement removes all rows from the Author_Book table where the Author_ID matches any Book_ID in the DELETED table. This ensures the Author_Book table does not contain references to books that no longer exist. This second step ensures the Author table does not contain "orphaned" authors (authors who are not associated with any books).

The DELETE statement removes all authors from the Author table where the Author_ID is not found in the Author_Book table.

Here are all the insertions we made in every table:

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database Execute

Object Explorer

Connect

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK)

Title (varchar)

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
-- Book
insert into Book (Book_ID, title, Publish_Date, publisher, Copies_available)
VALUES
(1, 'The Great Gatsby', '1925-04-10', 'Charles Scribners Sons', 5),
(2, '1984', '1949-06-08', 'Secker & Warburg', 3),
(3, 'To Kill a Mockingbird', '1960-07-11', 'J.B. Lippincott & Co.', 4),
(4, 'Moby-Dick', '1851-10-18', 'Harper & Brothers', 2),
(5, 'Pride and Prejudice', '1813-01-28', 'T. Egerton, Whitehall', 6),
(6, 'The Catcher in the Rye', '1951-07-16', 'Little, Brown and Company', 4),
(7, 'The Hobbit', '1937-09-21', 'George Allen & Unwin', 7),
(8, 'War and Peace', '1869-01-01', 'The Russian Messenger', 3),
(9, 'The Odyssey', '1900-01-01', 'Ancient Texts', 8),
(10, 'Crime and Punishment', '1866-01-01', 'The Russian Messenger', 5),
(11, 'The Grapes of Wrath', '1939-04-14', 'The Viking Press', 4),
(12, 'Wuthering Heights', '1847-12-01', 'Thomas Cautley Newby', 3),
(13, 'Frankenstein', '1818-01-01', 'Lackington, Hughes, Harding, Mavor & Jones', 6),
(14, 'Dracula', '1897-05-26', 'Archibald Constable and Company', 2),
(15, 'Anna Karenina', '1878-04-01', 'The Russian Messenger', 5),
(16, 'Brave New World', '1932-08-01', 'Chatto & Windus', 3),
(17, 'The Picture of Dorian Gray', '1890-07-01', 'Lippincotts Monthly Magazine', 4),
(18, 'Heart of Darkness', '1899-02-01', 'Blackwoods Magazine', 3),
(19, 'Les Misérables', '1862-04-03', 'A. Lacroix, Verboeckhoven & Cie', 7),
(20, 'The Divine Comedy', '1900-01-01', 'Ancient Texts', 5),
(21, 'The Lord of the Rings', '1954-07-29', 'George Allen & Unwin', 9),
(22, 'Ulysses', '1922-02-02', 'Sylvia Beach', 4),
(23, 'The Catcher in the Rye', '1951-07-16', 'Little, Brown and Company', 3),
(24, 'Catcher in the Rye', '1951-07-16', 'Little, Brown and Company', 5),
(25, 'The Chronicles of Narnia', '1950-10-16', 'Geoffrey Bles', 8),
(26, 'Harry Potter and the Philosophers Stone', '1997-06-26', 'George Allen & Unwin', 5),
(27, 'The Old Man and the Sea', '1952-09-01', 'Charles Scribners Sons', 4),
(28, 'Jane Eyre', '1847-10-16', 'Smith, Elder & Co.', 6);
```

100 %

Messages

Disconnected.

Ready

Ln 132 Col 1 Ch 1 INS

Type here to search

19°C عالم جزيئا

7:46 PM 12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database Execute

Object Explorer

Connect

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK)

Title (varchar)

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
--Writes
insert into Author_Book(Author_ID, Book_ID)
values
(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (8, 8), (9, 9), (10, 10),
(11, 11), (12, 12), (13, 13), (14, 14), (15, 15), (16, 16), (17, 17), (18, 18), (19, 19),
(21, 21), (22, 22), (23, 23), (24, 24), (25, 25), (26, 26), (27, 27), (28, 28);

--Genre
insert into Genre (Genre_ID, Name) values
(1, 'Fiction'), (2, 'Fantasy'), (3, 'Science Fiction'), (4, 'Mystery'),
(5, 'Romance'), (6, 'Historical Fiction'), (7, 'Thriller'), (8, 'Horror'),
(9, 'Adventure'), (10, 'Young Adult'), (11, 'Childrens Literature'),
(12, 'Biography'), (13, 'Autobiography'), (14, 'Memoir'), (15, 'Self-Help'),
(16, 'Non-Fiction'), (17, 'Classics'), (18, 'Humor'), (19, 'Poetry'),
(20, 'Drama'), (21, 'Graphic Novel'), (22, 'Short Stories'), (23, 'Contemporary'),
(24, 'Literary Fiction'), (25, 'Crime');

--Has Genre
insert into Has_Genre (book_id, genre_id) values
(1, 21),
(2, 1), (2, 24),
(3, 9), (3, 8),
(4, 8), (4, 5),
(5, 24), (5, 4),
(6, 14),
(7, 25), (7, 21),
(8, 9), (8, 24),
(9, 13), (9, 14),
(10, 19),
(11, 1),
(12, 4),
```

100 %

Messages

Disconnected.

Ready

Ln 132 Col 1 Ch 1 INS

Type here to search

19°C عالم جزيئا

7:47 PM 12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database Execute

Object Explorer

Connect

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

- System Databases
- Database Snapshots
- Library
- Library Database
 - Database Diagrams
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - Graph Tables
 - dbo.Author
 - dbo.Author_Book
 - dbo.Book
 - Columns
 - Book_ID (PK, i
 - Title (varchar(
 - Publish_Date
 - Copies_Availa
 - Publisher (var
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.Fine
 - dbo.Genre
 - dbo.Has_Genre

query.sql - not connected

```
(28, 'Jane Eyre', '1847-10-16', 'Smith, Elder & Co.', 6);

--Author
insert into Author (Author_ID, Author_Name, Nationality)
values
(1, 'F. Scott Fitzgerald', 'American'),
(2, 'George Orwell', 'British'),
(3, 'Harper Lee', 'American'),
(4, 'Herman Melville', 'American'),
(5, 'Jane Austen', 'British'),
(6, 'J.D. Salinger', 'American'),
(7, 'J.R.R. Tolkien', 'British'),
(8, 'Leo Tolstoy', 'Russian'),
(9, 'Homer', 'Greek'),
(10, 'Fyodor Dostoevsky', 'Russian'),
(11, 'John Steinbeck', 'American'),
(12, 'Emily Brontë', 'British'),
(13, 'Mary Shelley', 'British'),
(14, 'Bram Stoker', 'Irish'),
(15, 'Charles Dickens', 'British'),
(16, 'Mark Twain', 'American'),
(17, 'J.K. Rowling', 'British'),
(18, 'Gabriel García Márquez', 'Colombian'),
(19, 'Virginia Woolf', 'British'),
(20, 'Agatha Christie', 'British'),
(21, 'Isaac Asimov', 'American'),
(22, 'Arthur C. Clarke', 'British'),
(23, 'Philip K. Dick', 'American'),
(24, 'Ray Bradbury', 'American'),
(25, 'Kurt Vonnegut', 'American'),
(26, 'J.K. Rowling', 'British'),
```

100 %

Messages

Disconnected.

Ready Ln 132 Col 1 Ch 1 INS

Type here to search

19°C عانم حرتنا

7:46 PM 12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database

Object Explorer

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK, i

Title (varchar

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
--Members
insert into Member (Member_ID, Name, Registration_Date, Email_Address)
values
(1, 'John Doe', '2024-01-01', 'johndoe@example.com'),
(2, 'Jane Smith', '2024-01-03', 'janesmith@example.com'),
(3, 'Michael Johnson', '2024-01-05', 'michaelj@example.com'),
(4, 'Emily Brown', '2024-01-07', 'emilyb@example.com'),
(5, 'David Wilson', '2024-01-09', 'davidw@example.com'),
(6, 'Sarah Lee', '2024-01-12', 'sarahlee@example.com'),
(7, 'James Taylor', '2024-01-15', 'jamestaylor@example.com'),
(8, 'Laura Martinez', '2024-01-18', 'lauram@example.com'),
(9, 'Paul Anderson', '2024-01-21', 'paulanderson@example.com'),
(10, 'Olivia Thomas', '2024-01-25', 'oliviatt@example.com'),
(11, 'Daniel White', '2024-01-28', 'danielw@example.com'),
(12, 'Sophia Harris', '2024-02-01', 'sophiah@example.com'),
(13, 'Liam Clark', '2024-02-04', 'liamc@example.com'),
(14, 'Ethan Lewis', '2024-02-07', 'ethanl@example.com'),
(15, 'Ava Walker', '2024-02-10', 'avaw@example.com'),
(16, 'Mason Hall', '2024-02-13', 'masonh@example.com'),
(17, 'Isabella Allen', '2024-02-16', 'isabella@example.com'),
(18, 'Lucas Young', '2024-02-19', 'lucasy@example.com'),
(19, 'Charlotte King', '2024-02-22', 'charlottek@example.com'),
(20, 'Amelia Scott', '2024-02-25', 'amelias@example.com'),
(21, 'Benjamin Adams', '2024-02-28', 'benjamin@example.com'),
(22, 'Harper Baker', '2024-03-03', 'harperb@example.com'),
(23, 'William Gonzalez', '2024-03-06', 'williamg@example.com'),
(24, 'Ella Perez', '2024-03-09', 'ellap@example.com'),
(25, 'Jack Roberts', '2024-03-12', 'jackr@example.com'),
(26, 'Mia Martinez', '2024-03-15', 'miam@example.com'),
(27, 'Henry Thompson', '2024-03-18', 'henryt@example.com'),
```

100 %

Messages

Disconnected.

Ready

Ln 132 Col 1 Ch 1 INS

Type here to search

19°C غائم جزئيًا

ENG

7:47 PM

12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database

Object Explorer

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK, i

Title (varchar

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
(30, 'Chloe Jackson', '2024-03-27', 'chloej@example.com');

--MemberPhone
insert into Member_Phone (Member_ID, Phone_Number)
values
(1, '11234567890'), (1, '15012345678'),
(2, '12345678901'),
(3, '11298765432'), (3, '15023456789'),
(4, '12123456789'),
(5, '11234567891'), (5, '15034567890'),
(6, '12345678902'),
(7, '11234567892'), (7, '15045678901'),
(8, '12134567890'),
(9, '11234567893'), (9, '15056789012'),
(10, '12345678903'),
(11, '11234567894'), (11, '15067890123'),
(12, '12145678901'),
(13, '11234567895'), (13, '15078901234'),
(14, '12345678904'),
(15, '11234567896'), (15, '15089012345'),
(16, '12156789012'),
(17, '11234567897'), (17, '15090123456'),
(18, '12345678905'),
(19, '11234567898'), (19, '15091234567'),
(20, '12167890123'),
(21, '11234567899'),
(22, '15092345678'), (22, '12345678906'),
(23, '11234567900'),
(24, '15093456789'), (24, '12178901234'),
(25, '11234567901'), (25, '15094567890'),
(26, '12345678907'),
```

100 %

Messages

Disconnected.

Ready

Ln 132 Col 1 Ch 1 INS

Type here to search

19°C غائم جزئيًا

ENG

7:47 PM

12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database

Object Explorer

Connect

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK, i

Title (varchar(

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
--Staff
insert into Staff (Staff_ID, Name, Phone_Number, Hire_Date, Salary)
values
(1, 'Alice Turner', 7001234561, '2022-01-15', 4000.00),
(2, 'Brad Cooper', 7001234562, '2021-05-20', 5500.00),
(3, 'Cathy Roberts', 7001234563, '2023-03-10', 4000.00),
(4, 'David Morgan', 7001234564, '2019-07-25', 4000.00),
(5, 'Emma Sanchez', 7001234565, '2020-10-05', 4000.00);

-- Loan
insert into Loan (Loan_ID, Loan_Date, Return_Date, Member_ID, Staff_ID)
values
(1, '2024-01-01', '2024-01-15', 1, 1),
(2, '2024-01-03', '2024-01-17', 2, 2),
(3, '2024-01-05', '2024-01-19', 3, 3),
(4, '2024-01-07', '2024-01-21', 4, 1),
(5, '2024-01-09', '2024-01-23', 5, 2),
(6, '2024-01-12', '2024-01-26', 1, 3),
(7, '2024-01-15', '2024-01-29', 4, 1),
(8, '2024-01-18', '2024-02-01', 8, 2),
(9, '2024-01-21', '2024-02-04', 9, 3),
(10, '2024-01-25', '2024-02-08', 10, 1),
(11, '2024-02-01', '2024-02-15', 11, 2),
(12, '2024-02-07', '2024-02-21', 12, 3),
(13, '2024-02-13', '2024-02-27', 13, 1),
(14, '2024-02-19', '2024-03-04', 4, 2),
(15, '2024-02-25', '2024-03-10', 15, 3),
(16, '2024-03-03', '2024-03-17', 16, 1),
(17, '2024-03-09', '2024-03-23', 17, 2),
(18, '2024-03-15', '2024-03-29', 18, 3),
(19, '2024-03-21', '2024-04-04', 19, 1),
```

100 %

Messages

Disconnected.

Ready

Ln 132 Col 1 Ch 1 INS

Type here to search

19°C عالم جزيتا ENG 7:48 PM 12/25/2024

query.sql - not connected - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

Library Database

Object Explorer

Connect

DESKTOP-852ELFL\SQLEXPRESS (SC)

Databases

System Databases

Database Snapshots

Library Database

Database Diagrams

Tables

System Tables

FileTables

External Tables

Graph Tables

dbo.Author

dbo.Author_Book

dbo.Book

Columns

Book_ID (PK, i

Title (varchar(

Publish_Date

Copies_Availa

Publisher (var

Keys

Constraints

Triggers

Indexes

Statistics

dbo.Fine

dbo.Genre

dbo.Has_Genre

query.sql - not connected

```
--Loan_Includes
INSERT INTO Loan_Includes (Book_ID, Loan_ID)
VALUES
(1, 1),
(1, 5),
(2, 4),
(3, 3),
(3, 4),
(6, 5),
(7, 2),
(9, 1),
(10, 3);

-- Fine
insert into Fine (Fine_ID, Fine_Per_Day, Previous_Due_Date, Payment_Status, Due_Date, Loan_ID)
values
(1, 12.00, '2024-01-15', 'Paid', '2024-01-21', 1), -- 20 days overdue (6 days fined)
(2, 12.00, '2024-01-17', 'Pending', '2024-01-18', 2), -- 15 days overdue (1 day fined)
(3, 12.00, '2024-01-19', 'Paid', '2024-01-22', 3), -- 17 days overdue (3 days fined)
(4, 12.00, '2024-02-01', 'Pending', '2024-02-05', 8), -- 18 days overdue (4 days fined)
(5, 12.00, '2024-01-23', 'Paid', '2024-01-25', 10), -- 16 days overdue (2 days fined)
(6, 12.00, '2024-01-26', 'Pending', '2024-01-27', 11), -- 15 days overdue (1 day fined)
(7, 12.00, '2024-01-29', 'Paid', '2024-01-31', 12), -- 16 days overdue (2 days fined)
(8, 12.00, '2024-02-04', 'Pending', '2024-02-6', 15), -- 16 days overdue (2 days fined)
(9, 12.00, '2024-02-08', 'Paid', '2024-02-11', 16), -- 17 days overdue (3 days fined)
(10, 12.00, '2024-02-10', 'Pending', '2024-02-13', 17); -- 17 days overdue (3 days fined)
```

100 %

Messages

Disconnected.

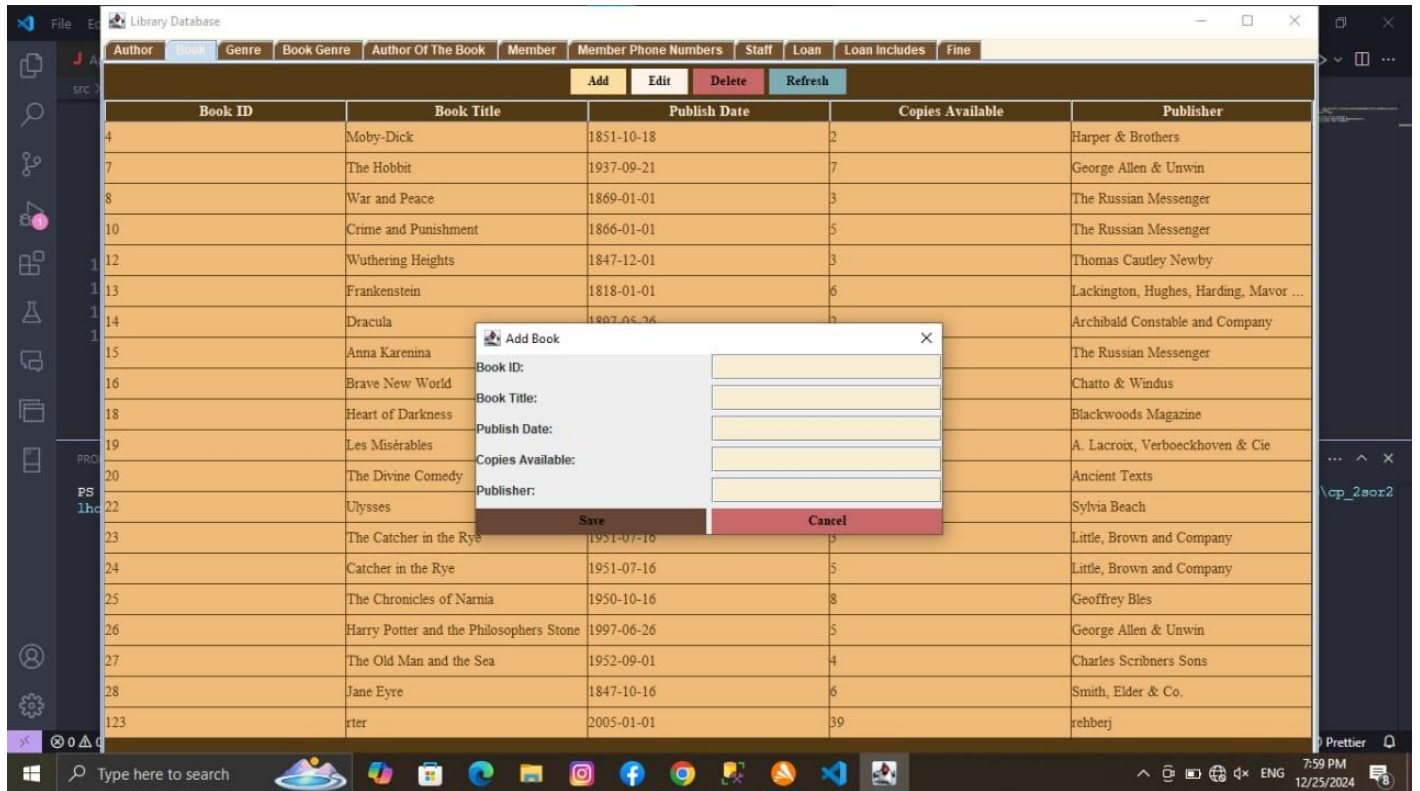
Ready

Ln 392 Col 12 Ch 9 INS

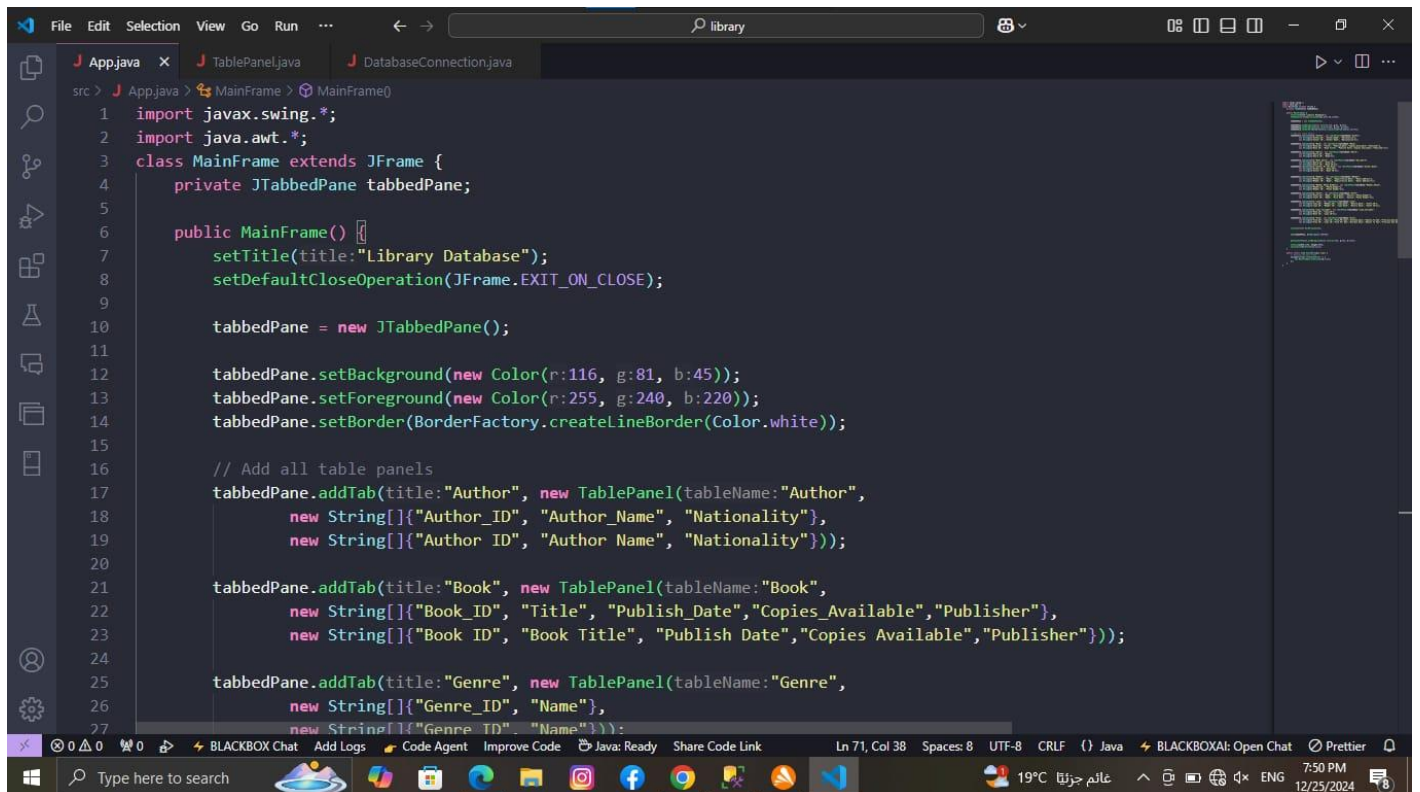
Type here to search

19°C عالم جزيتا ENG 7:48 PM 12/25/2024

GUI



GUI CODE



```
File Edit Selection View Go Run ... library
App.java x TablePanel.java DatabaseConnection.java
src > App.java > MainFrame > MainFrame()
3 class MainFrame extends JFrame {
6 public MainFrame() {
25 tabbedPane.addTab(title:"Genre", new TablePanel(tableName:"Genre",
26 new String[]{"Genre_ID", "Name"},
27 new String[]{"Genre ID", "Name"}));
28
29 tabbedPane.addTab(title:"Book Genre", new TablePanel(tableName:"Has_Genre",
30 new String[]{"genre_id", "book_id"},
31 new String[]{"Genre ID", "Book ID"}));
32
33 tabbedPane.addTab(title:"Author Of The Book", new TablePanel(tableName:"Author_Book",
34 new String[]{"Author_ID", "Book_ID"},
35 new String[]{"Author ID", "Book ID"}));
36
37 tabbedPane.addTab(title:"Member", new TablePanel(tableName:"Member",
38 new String[]{"Member_ID", "Name", "Registration_Date", "Email_Address"},
39 new String[]{"Member ID", "Name", "Registration Date", "Email Address"}));
40
41 tabbedPane.addTab(title:"Member Phone Numbers", new TablePanel(tableName:"Member_Phone",
42 new String[]{"Member_ID", "Phone_Number"},
43 new String[]{"Member ID", "Phone Number"}));
44
45 tabbedPane.addTab(title:"Staff", new TablePanel(tableName:"Staff",
46 new String[]{"Staff_ID", "Name", "Hire_Date", "Salary", "Phone_Number"},
47 new String[]{"Staff ID", "Name", "Hire Date", "Salary", "Phone Number"}));
48
49 tabbedPane.addTab(title:"Loan", new TablePanel(tableName:"Loan",
50 new String[]{"Loan_ID", "Member_ID", "Loan_Date", "Return_Date", "Staff_ID"},
51 new String[]{"Loan ID", "Member ID", "Loan Date", "Return Date", "Staff ID"}));
52
53 tabbedPane.addTab(title:"Loan Includes", new TablePanel(tableName:"Loan_Includes",
54 new String[]{"Book_ID", "Loan_ID"},
55 new String[]{"Book ID", "Loan ID"}));
56
57 tabbedPane.addTab(title:"Fine", new TablePanel(tableName:"Fine",
58 new String[]{"Fine_ID", "Loan_ID", "Fine_Per_Day", "Overdue_Days", "Amount_To_Pay", "Previous_Due_Date", "D"},
59 new String[]{"Fine ID", "Loan ID", "Fine Per Day", "Overdue Days", "Amount To Pay", "Previous Due Date", "D"});
60
61
62
63
64
65
66
67
68
69
70
71
72
Ln 71, Col 38 Spaces: 8 UTF-8 CRLF () Java BLACKBOXAI: Open Chat Prettier
```

```
File Edit Selection View Go Run ... library
App.java x TablePanel.java DatabaseConnection.java
src > App.java > MainFrame > MainFrame()
3 class MainFrame extends JFrame {
6 public MainFrame() {
48
49 tabbedPane.addTab(title:"Loan", new TablePanel(tableName:"Loan",
50 new String[]{"Loan_ID", "Member_ID", "Loan_Date", "Return_Date", "Staff_ID"},
51 new String[]{"Loan ID", "Member ID", "Loan Date", "Return Date", "Staff ID"}));
52
53 tabbedPane.addTab(title:"Loan Includes", new TablePanel(tableName:"Loan_Includes",
54 new String[]{"Book_ID", "Loan_ID"},
55 new String[]{"Book ID", "Loan ID"}));
56
57 tabbedPane.addTab(title:"Fine", new TablePanel(tableName:"Fine",
58 new String[]{"Fine_ID", "Loan_ID", "Fine_Per_Day", "Overdue_Days", "Amount_To_Pay", "Previous_Due_Date", "D"},
59 new String[]{"Fine ID", "Loan ID", "Fine Per Day", "Overdue Days", "Amount To Pay", "Previous Due Date", "D"});
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
Ln 71, Col 38 Spaces: 8 UTF-8 CRLF () Java BLACKBOXAI: Open Chat Prettier
```

```
File Edit Selection View Go Run ... library
App.java x TablePanel.java DatabaseConnection.java
src > App.java > MainFrame > MainFrame()
3 class MainFrame extends JFrame {
6 public MainFrame() {
62     setLayout(new BorderLayout());
63
64
65     add(tabbedPane, BorderLayout.CENTER);
66
67
68     getContentPane().setBackground(new Color(r:248, g:244, b:225));
69
70     setSize(width:1200, height:800);
71     setLocationRelativeTo(c:null);
72
73
74 Run | Debug
74 public static void main(String[] args) {
75     // Running the main frame
76     SwingUtilities.invokeLater(() -> {
77         new MainFrame().setVisible(b:true);
78     });
79 }
80 }
81

Type here to search BLACKBOX Chat Add Logs Code Agent Improve Code Java: Ready Share Code Link Ln 71, Col 38 Spaces: 8 UTF-8 CRLF () Java BLACKBOXAI: Open Chat Prettier 19°C غائم جزئيًا 7:51 PM 12/25/2024
```

```
File Edit Selection View Go Run ... library
App.java x TablePanel.java DatabaseConnection.java
src > TablePanel.java > TablePanel > showUpdateDialog()
1 import javax.swing.*.*;
2 import javax.swing.table.DefaultTableModel;
3 import javax.swing.table.JTableHeader;
4 import java.awt.*.*;
5 import java.sql.*.*;
6
7 public class TablePanel extends JPanel {
8     private final String tableName;
9     private final String[] columnNames;
10    private final String[] displayNames;
11    private JTable table;
12
13    public TablePanel(String tableName, String[] columnNames, String[] displayNames) {
14        this.tableName = tableName;
15        this.columnNames = columnNames;
16        this.displayNames = displayNames;
17
18        setLayout(new BorderLayout());
19        initializeComponents();
20    }
21
22    private void initializeComponents() {
23        // Create table model
24        DefaultTableModel model = new DefaultTableModel(columnNames, rowCount:0);
25        table = new JTable(model);
26        table.setBackground(new Color(r:240, g:187, b:120));
27        table.setForeground(new Color(r:84, g:58, b:20));
28    }
29
30    private void showUpdateDialog() {
31        // Show dialog to update table data
32        // ...
33    }
34
35    private void refreshTable() {
36        // Refresh table data
37        // ...
38    }
39
40    private void deleteRow() {
41        // Delete row from table
42        // ...
43    }
44
45    private void addRow() {
46        // Add row to table
47        // ...
48    }
49
50    private void editRow() {
51        // Edit row in table
52        // ...
53    }
54
55    private void searchTable() {
56        // Search table
57        // ...
58    }
59
60    private void filterTable() {
61        // Filter table
62        // ...
63    }
64
65    private void sortTable() {
66        // Sort table
67        // ...
68    }
69
70    private void exportTable() {
71        // Export table
72        // ...
73    }
74
75    private void importTable() {
76        // Import table
77        // ...
78    }
79
80    private void printTable() {
81        // Print table
82        // ...
83    }
84
85    private void saveTable() {
86        // Save table
87        // ...
88    }
89
90    private void loadTable() {
91        // Load table
92        // ...
93    }
94
95    private void resetTable() {
96        // Reset table
97        // ...
98    }
99
100   private void clearTable() {
101       // Clear table
102       // ...
103   }
104
105   private void showTable() {
106       // Show table
107       // ...
108   }
109
110   private void hideTable() {
111       // Hide table
112       // ...
113   }
114
115   private void toggleTable() {
116       // Toggle table
117       // ...
118   }
119
120   private void showTableDetails() {
121       // Show table details
122       // ...
123   }
124
125   private void hideTableDetails() {
126       // Hide table details
127       // ...
128   }
129
130   private void toggleTableDetails() {
131       // Toggle table details
132       // ...
133   }
134
135   private void showTableSummary() {
136       // Show table summary
137       // ...
138   }
139
140   private void hideTableSummary() {
141       // Hide table summary
142       // ...
143   }
144
145   private void toggleTableSummary() {
146       // Toggle table summary
147       // ...
148   }
149
150   private void showTableStats() {
151       // Show table stats
152       // ...
153   }
154
155   private void hideTableStats() {
156       // Hide table stats
157       // ...
158   }
159
160   private void toggleTableStats() {
161       // Toggle table stats
162       // ...
163   }
164
165   private void showTableReport() {
166       // Show table report
167       // ...
168   }
169
170   private void hideTableReport() {
171       // Hide table report
172       // ...
173   }
174
175   private void toggleTableReport() {
176       // Toggle table report
177       // ...
178   }
179
180   private void showTableLogs() {
181       // Show table logs
182       // ...
183   }
184
185   private void hideTableLogs() {
186       // Hide table logs
187       // ...
188   }
189
190   private void toggleTableLogs() {
191       // Toggle table logs
192       // ...
193   }
194
195   private void showTableHelp() {
196       // Show table help
197       // ...
198   }
199
200   private void hideTableHelp() {
201       // Hide table help
202       // ...
203   }
204
205   private void toggleTableHelp() {
206       // Toggle table help
207       // ...
208   }
209
210   private void showTableAbout() {
211       // Show table about
212       // ...
213   }
214
215   private void hideTableAbout() {
216       // Hide table about
217       // ...
218   }
219
220   private void toggleTableAbout() {
221       // Toggle table about
222       // ...
223   }
224
225   private void showTableVersion() {
226       // Show table version
227       // ...
228   }
229
230   private void hideTableVersion() {
231       // Hide table version
232       // ...
233   }
234
235   private void toggleTableVersion() {
236       // Toggle table version
237       // ...
238   }
239
240   private void showTableLicense() {
241       // Show table license
242       // ...
243   }
244
245   private void hideTableLicense() {
246       // Hide table license
247       // ...
248   }
249
250   private void toggleTableLicense() {
251       // Toggle table license
252       // ...
253   }
254
255   private void showTablePrivacy() {
256       // Show table privacy
257       // ...
258   }
259
260   private void hideTablePrivacy() {
261       // Hide table privacy
262       // ...
263   }
264
265   private void toggleTablePrivacy() {
266       // Toggle table privacy
267       // ...
268   }
269
270   private void showTableTerms() {
271       // Show table terms
272       // ...
273   }
274
275   private void hideTableTerms() {
276       // Hide table terms
277       // ...
278   }
279
280   private void toggleTableTerms() {
281       // Toggle table terms
282       // ...
283   }
284
285   private void showTableDisclaimer() {
286       // Show table disclaimer
287       // ...
288   }
289
290   private void hideTableDisclaimer() {
291       // Hide table disclaimer
292       // ...
293   }
294
295   private void toggleTableDisclaimer() {
296       // Toggle table disclaimer
297       // ...
298   }
299
300   private void showTableNotice() {
301       // Show table notice
302       // ...
303   }
304
305   private void hideTableNotice() {
306       // Hide table notice
307       // ...
308   }
309
310   private void toggleTableNotice() {
311       // Toggle table notice
312       // ...
313   }
314
315   private void showTableWarning() {
316       // Show table warning
317       // ...
318   }
319
320   private void hideTableWarning() {
321       // Hide table warning
322       // ...
323   }
324
325   private void toggleTableWarning() {
326       // Toggle table warning
327       // ...
328   }
329
330   private void showTableError() {
331       // Show table error
332       // ...
333   }
334
335   private void hideTableError() {
336       // Hide table error
337       // ...
338   }
339
340   private void toggleTableError() {
341       // Toggle table error
342       // ...
343   }
344
345   private void showTableInfo() {
346       // Show table info
347       // ...
348   }
349
350   private void hideTableInfo() {
351       // Hide table info
352       // ...
353   }
354
355   private void toggleTableInfo() {
356       // Toggle table info
357       // ...
358   }
359
360   private void showTableStatus() {
361       // Show table status
362       // ...
363   }
364
365   private void hideTableStatus() {
366       // Hide table status
367       // ...
368   }
369
370   private void toggleTableStatus() {
371       // Toggle table status
372       // ...
373   }
374
375   private void showTableProgress() {
376       // Show table progress
377       // ...
378   }
379
380   private void hideTableProgress() {
381       // Hide table progress
382       // ...
383   }
384
385   private void toggleTableProgress() {
386       // Toggle table progress
387       // ...
388   }
389
390   private void showTableLoading() {
391       // Show table loading
392       // ...
393   }
394
395   private void hideTableLoading() {
396       // Hide table loading
397       // ...
398   }
399
400   private void toggleTableLoading() {
401       // Toggle table loading
402       // ...
403   }
404
405   private void showTableBusy() {
406       // Show table busy
407       // ...
408   }
409
410   private void hideTableBusy() {
411       // Hide table busy
412       // ...
413   }
414
415   private void toggleTableBusy() {
416       // Toggle table busy
417       // ...
418   }
419
420   private void showTableIdle() {
421       // Show table idle
422       // ...
423   }
424
425   private void hideTableIdle() {
426       // Hide table idle
427       // ...
428   }
429
430   private void toggleTableIdle() {
431       // Toggle table idle
432       // ...
433   }
434
435   private void showTableReady() {
436       // Show table ready
437       // ...
438   }
439
440   private void hideTableReady() {
441       // Hide table ready
442       // ...
443   }
444
445   private void toggleTableReady() {
446       // Toggle table ready
447       // ...
448   }
449
450   private void showTableAvailable() {
451       // Show table available
452       // ...
453   }
454
455   private void hideTableAvailable() {
456       // Hide table available
457       // ...
458   }
459
460   private void toggleTableAvailable() {
461       // Toggle table available
462       // ...
463   }
464
465   private void showTableOffline() {
466       // Show table offline
467       // ...
468   }
469
470   private void hideTableOffline() {
471       // Hide table offline
472       // ...
473   }
474
475   private void toggleTableOffline() {
476       // Toggle table offline
477       // ...
478   }
479
480   private void showTableOnline() {
481       // Show table online
482       // ...
483   }
484
485   private void hideTableOnline() {
486       // Hide table online
487       // ...
488   }
489
490   private void toggleTableOnline() {
491       // Toggle table online
492       // ...
493   }
494
495   private void showTableConnected() {
496       // Show table connected
497       // ...
498   }
499
500   private void hideTableConnected() {
501       // Hide table connected
502       // ...
503   }
504
505   private void toggleTableConnected() {
506       // Toggle table connected
507       // ...
508   }
509
510   private void showTableDisconnected() {
511       // Show table disconnected
512       // ...
513   }
514
515   private void hideTableDisconnected() {
516       // Hide table disconnected
517       // ...
518   }
519
520   private void toggleTableDisconnected() {
521       // Toggle table disconnected
522       // ...
523   }
524
525   private void showTableAuthenticated() {
526       // Show table authenticated
527       // ...
528   }
529
530   private void hideTableAuthenticated() {
531       // Hide table authenticated
532       // ...
533   }
534
535   private void toggleTableAuthenticated() {
536       // Toggle table authenticated
537       // ...
538   }
539
540   private void showTableUnauthenticated() {
541       // Show table unauthenticated
542       // ...
543   }
544
545   private void hideTableUnauthenticated() {
546       // Hide table unauthenticated
547       // ...
548   }
549
550   private void toggleTableUnauthenticated() {
551       // Toggle table unauthenticated
552       // ...
553   }
554
555   private void showTableAuthorized() {
556       // Show table authorized
557       // ...
558   }
559
560   private void hideTableAuthorized() {
561       // Hide table authorized
562       // ...
563   }
564
565   private void toggleTableAuthorized() {
566       // Toggle table authorized
567       // ...
568   }
569
570   private void showTableUnauthorized() {
571       // Show table unauthorized
572       // ...
573   }
574
575   private void hideTableUnauthorized() {
576       // Hide table unauthorized
577       // ...
578   }
579
580   private void toggleTableUnauthorized() {
581       // Toggle table unauthorized
582       // ...
583   }
584
585   private void showTableAdmin() {
586       // Show table admin
587       // ...
588   }
589
590   private void hideTableAdmin() {
591       // Hide table admin
592       // ...
593   }
594
595   private void toggleTableAdmin() {
596       // Toggle table admin
597       // ...
598   }
599
600   private void showTableUser() {
601       // Show table user
602       // ...
603   }
604
605   private void hideTableUser() {
606       // Hide table user
607       // ...
608   }
609
610   private void toggleTableUser() {
611       // Toggle table user
612       // ...
613   }
614
615   private void showTableGuest() {
616       // Show table guest
617       // ...
618   }
619
620   private void hideTableGuest() {
621       // Hide table guest
622       // ...
623   }
624
625   private void toggleTableGuest() {
626       // Toggle table guest
627       // ...
628   }
629
630   private void showTableVisitor() {
631       // Show table visitor
632       // ...
633   }
634
635   private void hideTableVisitor() {
636       // Hide table visitor
637       // ...
638   }
639
640   private void toggleTableVisitor() {
641       // Toggle table visitor
642       // ...
643   }
644
645   private void showTableAnonymous() {
646       // Show table anonymous
647       // ...
648   }
649
650   private void hideTableAnonymous() {
651       // Hide table anonymous
652       // ...
653   }
654
655   private void toggleTableAnonymous() {
656       // Toggle table anonymous
657       // ...
658   }
659
660   private void showTableAuthenticatedUser() {
661       // Show table authenticated user
662       // ...
663   }
664
665   private void hideTableAuthenticatedUser() {
666       // Hide table authenticated user
667       // ...
668   }
669
670   private void toggleTableAuthenticatedUser() {
671       // Toggle table authenticated user
672       // ...
673   }
674
675   private void showTableUnauthenticatedUser() {
676       // Show table unauthenticated user
677       // ...
678   }
679
680   private void hideTableUnauthenticatedUser() {
681       // Hide table unauthenticated user
682       // ...
683   }
684
685   private void toggleTableUnauthenticatedUser() {
686       // Toggle table unauthenticated user
687       // ...
688   }
689
690   private void showTableAuthorizedUser() {
691       // Show table authorized user
692       // ...
693   }
694
695   private void hideTableAuthorizedUser() {
696       // Hide table authorized user
697       // ...
698   }
699
700   private void toggleTableAuthorizedUser() {
701       // Toggle table authorized user
702       // ...
703   }
704
705   private void showTableUnauthorizedUser() {
706       // Show table unauthorized user
707       // ...
708   }
709
710   private void hideTableUnauthorizedUser() {
711       // Hide table unauthorized user
712       // ...
713   }
714
715   private void toggleTableUnauthorizedUser() {
716       // Toggle table unauthorized user
717       // ...
718   }
719
720   private void showTableAdminUser() {
721       // Show table admin user
722       // ...
723   }
724
725   private void hideTableAdminUser() {
726       // Hide table admin user
727       // ...
728   }
729
730   private void toggleTableAdminUser() {
731       // Toggle table admin user
732       // ...
733   }
734
735   private void showTableUserUser() {
736       // Show table user user
737       // ...
738   }
739
740   private void hideTableUserUser() {
741       // Hide table user user
742       // ...
743   }
744
745   private void toggleTableUserUser() {
746       // Toggle table user user
747       // ...
748   }
749
750   private void showTableGuestUser() {
751       // Show table guest user
752       // ...
753   }
754
755   private void hideTableGuestUser() {
756       // Hide table guest user
757       // ...
758   }
759
760   private void toggleTableGuestUser() {
761       // Toggle table guest user
762       // ...
763   }
764
765   private void showTableVisitorUser() {
766       // Show table visitor user
767       // ...
768   }
769
770   private void hideTableVisitorUser() {
771       // Hide table visitor user
772       // ...
773   }
774
775   private void toggleTableVisitorUser() {
776       // Toggle table visitor user
777       // ...
778   }
779
780   private void showTableAnonymousUser() {
781       // Show table anonymous user
782       // ...
783   }
784
785   private void hideTableAnonymousUser() {
786       // Hide table anonymous user
787       // ...
788   }
789
790   private void toggleTableAnonymousUser() {
791       // Toggle table anonymous user
792       // ...
793   }
794
795   private void showTableAuthenticatedUserUser() {
796       // Show table authenticated user user
797       // ...
798   }
799
800   private void hideTableAuthenticatedUserUser() {
801       // Hide table authenticated user user
802       // ...
803   }
804
805   private void toggleTableAuthenticatedUserUser() {
806       // Toggle table authenticated user user
807       // ...
808   }
809
810   private void showTableUnauthenticatedUserUser() {
811       // Show table unauthenticated user user
812       // ...
813   }
814
815   private void hideTableUnauthenticatedUserUser() {
816       // Hide table unauthenticated user user
817       // ...
818   }
819
820   private void toggleTableUnauthenticatedUserUser() {
821       // Toggle table unauthenticated user user
822       // ...
823   }
824
825   private void showTableAuthorizedUserUser() {
826       // Show table authorized user user
827       // ...
828   }
829
830   private void hideTableAuthorizedUserUser() {
831       // Hide table authorized user user
832       // ...
833   }
834
835   private void toggleTableAuthorizedUserUser() {
836       // Toggle table authorized user user
837       // ...
838   }
839
840   private void showTableUnauthorizedUserUser() {
841       // Show table unauthorized user user
842       // ...
843   }
844
845   private void hideTableUnauthorizedUserUser() {
846       // Hide table unauthorized user user
847       // ...
848   }
849
850   private void toggleTableUnauthorizedUserUser() {
851       // Toggle table unauthorized user user
852       // ...
853   }
854
855   private void showTableAdminUserUser() {
856       // Show table admin user user
857       // ...
858   }
859
860   private void hideTableAdminUserUser() {
861       // Hide table admin user user
862       // ...
863   }
864
865   private void toggleTableAdminUserUser() {
866       // Toggle table admin user user
867       // ...
868   }
869
870   private void showTableUserUserUser() {
871       // Show table user user user
872       // ...
873   }
874
875   private void hideTableUserUserUser() {
876       // Hide table user user user
877       // ...
878   }
879
880   private void toggleTableUserUserUser() {
881       // Toggle table user user user
882       // ...
883   }
884
885   private void showTableGuestUserUser() {
886       // Show table guest user user
887       // ...
888   }
889
890   private void hideTableGuestUserUser() {
891       // Hide table guest user user
892       // ...
893   }
894
895   private void toggleTableGuestUserUser() {
896       // Toggle table guest user user
897       // ...
898   }
899
900   private void showTableVisitorUserUser() {
901       // Show table visitor user user
902       // ...
903   }
904
905   private void hideTableVisitorUserUser() {
906       // Hide table visitor user user
907       // ...
908   }
909
910   private void toggleTableVisitorUserUser() {
911       // Toggle table visitor user user
912       // ...
913   }
914
915   private void showTableAnonymousUserUser() {
916       // Show table anonymous user user
917       // ...
918   }
919
920   private void hideTableAnonymousUserUser() {
921       // Hide table anonymous user user
922       // ...
923   }
924
925   private void toggleTableAnonymousUserUser() {
926       // Toggle table anonymous user user
927       // ...
928   }
929
930   private void showTableAuthenticatedUserUserUser() {
931       // Show table authenticated user user user
932       // ...
933   }
934
935   private void hideTableAuthenticatedUserUserUser() {
936       // Hide table authenticated user user user
937       // ...
938   }
939
940   private void toggleTableAuthenticatedUserUserUser() {
941       // Toggle table authenticated user user user
942       // ...
943   }
944
945   private void showTableUnauthenticatedUserUserUser() {
946       // Show table unauthenticated user user user
947       // ...
948   }
949
950   private void hideTableUnauthenticatedUserUserUser() {
951       // Hide table unauthenticated user user user
952       // ...
953   }
954
955   private void toggleTableUnauthenticatedUserUserUser() {
956       // Toggle table unauthenticated user user user
957       // ...
958   }
959
960   private void showTableAuthorizedUserUserUser() {
961       // Show table authorized user user user
962       // ...
963   }
964
965   private void hideTableAuthorizedUserUserUser() {
966       // Hide table authorized user user user
967       // ...
968   }
969
970   private void toggleTableAuthorizedUserUserUser() {
971       // Toggle table authorized user user user
972       // ...
973   }
974
975   private void showTableUnauthorizedUserUserUser() {
976       // Show table unauthorized user user user
977       // ...
978   }
979
980   private void hideTableUnauthorizedUserUserUser() {
981       // Hide table unauthorized user user user
982       // ...
983   }
984
985   private void toggleTableUnauthorizedUserUserUser() {
986       // Toggle table unauthorized user user user
987       // ...
988   }
989
990   private void showTableAdminUserUserUser() {
991       // Show table admin user user user
992       // ...
993   }
994
995   private void hideTableAdminUserUserUser() {
996       // Hide table admin user user user
997       // ...
998   }
999
1000  private void toggleTableAdminUserUserUser() {
1001      // Toggle table admin user user user
1002      // ...
1003  }
1004
1005  private void showTableUserUserUserUser() {
1006      // Show table user user user user
1007      // ...
1008  }
1009
1010  private void hideTableUserUserUserUser() {
1011      // Hide table user user user user
1012      // ...
1013  }
1014
1015  private void toggleTableUserUserUserUser() {
1016      // Toggle table user user user user
1017      // ...
1018  }
1019
1020  private void showTableGuestUserUserUser() {
1021      // Show table guest user user user
1022      // ...
1023  }
1024
1025  private void hideTableGuestUserUserUser() {
1026      // Hide table guest user user user
1027      // ...
1028  }
1029
1030  private void toggleTableGuestUserUserUser() {
1031      // Toggle table guest user user user
1032      // ...
1033  }
1034
1035  private void showTableVisitorUserUserUser() {
1036      // Show table visitor user user user
1037      // ...
1038  }
1039
1040  private void hideTableVisitorUserUserUser() {
1041      // Hide table visitor user user user
1042      // ...
1043  }
1044
1045  private void toggleTableVisitorUserUserUser() {
1046      // Toggle table visitor user user user
1047      // ...
1048  }
1049
1050  private void showTableAnonymousUserUserUser() {
1051      // Show table anonymous user user user
1052      // ...
1053  }
1054
1055  private void hideTableAnonymousUserUserUser() {
1056      // Hide table anonymous user user user
1057      // ...
1058  }
1059
1060  private void toggleTableAnonymousUserUserUser() {
1061      // Toggle table anonymous user user user
1062      // ...
1063  }
1064
1065  private void showTableAuthenticatedUserUserUserUser() {
1066      // Show table authenticated user user user user
1067      // ...
1068  }
1069
1070  private void hideTableAuthenticatedUserUserUserUser() {
1071      // Hide table authenticated user user user user
1072      // ...
1073  }
1074
1075  private void toggleTableAuthenticatedUserUserUserUser() {
1076      // Toggle table authenticated user user user user
1077      // ...
1078  }
1079
1080  private void showTableUnauthenticatedUserUserUserUser() {
1081      // Show table unauthenticated user user user user
1082      // ...
1083  }
1084
1085  private void hideTableUnauthenticatedUserUserUserUser() {
1086      // Hide table unauthenticated user user user user
1087      // ...
1088  }
1089
1090  private void toggleTableUnauthenticatedUserUserUserUser() {
1091      // Toggle table unauthenticated user user user user
1092      // ...
1093  }
1094
1095  private void showTableAuthorizedUserUserUserUser() {
1096      // Show table authorized user user user user
1097      // ...
1098  }
1099
1100  private void hideTableAuthorizedUserUserUserUser() {
1101      // Hide table authorized user user user user
1102      // ...
1103  }
1104
1105  private void toggleTableAuthorizedUserUserUserUser() {
1106      // Toggle table authorized user user user user
1107      // ...
1108  }
1109
1110  private void showTableUnauthorizedUserUserUserUser() {
1111      // Show table unauthorized user user user user
1112      // ...
1113  }
1114
1115  private void hideTableUnauthorizedUserUserUserUser() {
1116      // Hide table unauthorized user user user user
1117      // ...
1118  }
1119
1120  private void toggleTableUnauthorizedUserUserUserUser() {
1121      // Toggle table unauthorized user user user user
1122      // ...
1123  }
1124
1125  private void showTableAdminUserUserUserUser() {
1126      // Show table admin user user user user
1127      // ...
1128  }
1129
1130  private void hideTableAdminUserUserUserUser() {
1131      // Hide table admin user user user user
1132      // ...
1133  }
1134
1135  private void toggleTableAdminUserUserUserUser() {
1136      // Toggle table admin user user user user
1137      // ...
1138  }
1139
1140  private void showTableUserUserUserUserUser() {
1141      // Show table user user user user user
1142      // ...
1143  }
1144
1145  private void hideTableUserUserUserUserUser() {
1146      // Hide table user user user user user
1147      // ...
1148  }
1149
1150  private void toggleTableUserUserUserUserUser() {
1151      // Toggle table user user user user user
1152      // ...
1153  }
1154
1155  private void showTableGuestUserUserUserUser() {
1156      // Show table guest user user user user
1157      // ...
1158  }
1159
1160  private void hideTableGuestUserUserUserUser() {
1161      // Hide table guest user user user user
1162      // ...
1163  }
1164
1165  private void toggleTableGuestUserUserUserUser() {
1166      // Toggle table guest user user user user
1167      // ...
1168  }
1169
1170  private void showTableVisitorUserUserUserUser() {
1171      // Show table visitor user user user user
1172      // ...
1173  }
1174
1175  private void hideTableVisitorUserUserUserUser() {
1176      // Hide table visitor user user user user
1177      // ...
1178  }
1179
1180  private void toggleTableVisitorUserUserUserUser() {
1181      // Toggle table visitor user user user user
1182      // ...
1183  }
1184
1185  private void showTableAnonymousUserUserUserUser() {
1186      // Show table anonymous user user user user
1187      // ...
1188  }
1189
1190  private void hideTableAnonymousUserUserUserUser() {
1191      // Hide table anonymous user user user user
1192      // ...
1193  }
1194
1195  private void toggleTableAnonymousUserUserUserUser() {
1196      // Toggle table anonymous user user user user
1197      // ...
1198  }
1199
1200  private void showTableAuthenticatedUserUserUserUserUser() {
1201      // Show table authenticated user user user user user
1202      // ...
1203  }
1204
1205  private void hideTableAuthenticatedUserUserUserUserUser() {
1206      // Hide table authenticated user user user user user
1207      // ...
1208  }
1209
1210  private void toggleTableAuthenticatedUserUserUserUserUser() {
1211      // Toggle table authenticated user user user user user
1212      // ...
1213  }
1214
1215  private void showTableUnauthenticatedUserUserUserUserUser() {
1216      // Show table unauthenticated user user user user user
1217      // ...
1218  }
1219
1220  private void hideTableUnauthenticatedUserUserUserUserUser() {
1221      // Hide table unauthenticated user user user user user
1222      // ...
1223  }
1224
1225  private void toggleTableUnauthenticatedUserUserUserUserUser() {
1226      // Toggle table unauthenticated user user user user user
1227      // ...
1228  }
1229
1230  private void showTableAuthorizedUserUserUserUserUser() {
1231      // Show table authorized user user user user user
1232      // ...
1233  }
1234
1235  private void hideTableAuthorizedUserUserUserUserUser() {
1236      // Hide table authorized user user user user user
1237      // ...
1238  }
1239
1240  private void toggleTableAuthorizedUserUserUserUserUser() {
1241      // Toggle table authorized user user user user user
1242      // ...
1243  }
1244
1245  private void showTableUnauthorizedUserUserUserUserUser() {
1246      // Show table unauthorized user user user user user
1247      // ...
1248  }
1249
1250  private void hideTableUnauthorizedUserUserUserUserUser() {
1251      // Hide table unauthorized user user user user user
1252      // ...
1253  }
1254
1255  private void toggleTableUnauthorizedUserUserUserUserUser() {
1256      // Toggle table unauthorized user user user user user
1257      // ...
1258  }
1259
1260  private void showTableAdminUserUserUserUserUser() {
1261      // Show table admin user user user user user
1262      // ...
1263  }
1264
1265  private void hideTableAdminUserUserUserUserUser() {
1266      // Hide table admin user user user user user
1267      // ...
1268  }
1269
1270  private void toggleTableAdminUserUserUserUserUser() {
1271      // Toggle table admin user user user user user
1272      // ...
1273  }
1274
1275  private void showTableUserUserUserUserUserUser() {
1276      // Show table user user user user user user
1277      // ...
1278  }
1279
1280  private void hideTableUserUserUserUserUserUser() {
1281      // Hide table user user user user user user
1282      // ...
1283  }
1284
1285  private void toggleTableUserUserUserUserUserUser() {
1286      // Toggle table user user user user user user
1287      // ...
1288  }
1289
1290  private void showTableGuestUserUserUserUserUser() {
1291      // Show table guest user user user user user
1292      // ...
1293  }
1294
1295  private void hideTableGuestUserUserUserUserUser() {
1296      // Hide table guest user user user user user
1297      // ...
1298  }
1299
1300  private void toggleTableGuestUserUserUserUserUser() {
1301      // Toggle table guest user user user user user
1302      // ...
1303  }
1304
1305  private void showTableVisitorUserUserUserUserUser() {
1306      // Show table visitor user user user user user
1307      // ...
1308  }
1309
1310  private void hideTableVisitorUserUserUserUserUser() {
1311      // Hide table visitor user user user user user
1312      // ...
1313  }
1314
1315  private void toggleTableVisitorUserUserUserUserUser() {
1316      // Toggle table visitor user user user user user
1317      // ...
1318  }
1319
1320  private void showTableAnonymousUserUserUserUserUser() {
1321      // Show table anonymous user user user user user
1322      // ...
1323  }
1324
1325  private void hideTableAnonymousUserUserUserUserUser() {
1326      // Hide table anonymous user user user user user
1327      // ...
1328  }
1329
1330  private void toggleTableAnonymousUserUserUserUserUser() {
1331      // Toggle table anonymous user user user user user
1332      // ...
1333  }
1334
1335  private void showTableAuthenticatedUserUserUserUserUserUser() {
1336      // Show table authenticated user user user user user user
1337      // ...
1338  }
1339
1340  private void hideTableAuthenticatedUserUserUserUserUserUser() {
1341      // Hide table authenticated user user user user user user
1342      // ...
1343  }
1344
1345  private void toggleTableAuthenticatedUserUserUserUserUserUser() {
1346      // Toggle table authenticated user user user user user user
1347      // ...
1348  }
1349
1350  private void showTableUnauthenticatedUserUserUserUserUserUser() {
1351      // Show table unauthenticated user user user user user user
1352      // ...
1353  }
1354
1355  private void hideTableUnauthenticatedUserUserUserUserUserUser() {
1356      // Hide table unauthenticated user user user user user user
1357      // ...
1358  }
1359
1360  private void toggleTableUnauthenticatedUserUserUserUserUserUser() {
1361      // Toggle table unauthenticated user user user user user user
1362      // ...
1363  }
1364
1365  private void showTableAuthorizedUserUserUserUserUserUser() {
1366      // Show table authorized user user user user user user
1367      // ...
1368  }
1369
1370  private void hideTableAuthorizedUserUserUserUserUserUser() {
1371      // Hide table authorized user user user user user user
1372      // ...
1373  }
1374
1375  private void toggleTableAuthorizedUserUserUserUserUserUser() {
1376      // Toggle table authorized user user user user user user
1377      // ...
1378  }
1379
1380  private void showTableUnauthorizedUserUserUserUserUserUser() {
1381      // Show table unauthorized user user user user user user
1382      // ...
1383  }
1384
1385  private void hideTableUnauthorizedUserUserUserUserUserUser() {
1386      // Hide table unauthorized user user user user user user
1387      // ...
1388  }
1389
1390  private void toggleTableUnauthorizedUserUserUserUserUserUser() {
1391      // Toggle table unauthorized user user user user user user
1392      // ...
1393  }
1394
1395  private void showTableAdminUserUserUserUserUserUser() {
1396      // Show table admin user user user user user user
1397      // ...
13
```



```
File Edit Selection View Go Run ... library
J App.java J TablePanel.java X DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
22 private void initializeComponents() {
23     table.setBackground(new Color(r:240, g:187, b:120));
27     table.setForeground(new Color(r:84, g:58, b:20));
28     table.setSelectionBackground(new Color(r:84, g:58, b:20));
29     table.setSelectionForeground(new Color(r:240, g:187, b:120));
30     table.setGridColor(new Color(r:84, g:58, b:20));
31     table.setFont(new Font(name:"Times new roman", Font.BOLD, size:14));
32     table.setRowHeight(rowHeight:30);
33
34     table.setOpaque(isOpaque:true);
35
36     JTableHeader tableHeader = table.getTableHeader();
37     tableHeader.setBackground(new Color(r:84, g:58, b:20));
38     tableHeader.setForeground(new Color(r:255, g:240, b:220));
39     tableHeader.setFont(new Font(name:"Times new roman", Font.BOLD, size:14));
40     tableHeader.setReorderingAllowed(reorderingAllowed:false);
41     tableHeader.setOpaque(isOpaque:true);
42
43     table.setDefaultRenderer(columnClass:Object.class, (table, value, isSelected, hasFocus, row, column) -> {
44         JLabel label = new JLabel(value == null || value.toString().trim().isEmpty() ? "" : value.toString());
45         label.setOpaque(isOpaque:true);
46         label.setBackground(isSelected ? table.getSelectionBackground() : table.getBackground());
47         label.setForeground(isSelected ? table.getSelectionForeground() : table.getForeground());
48         label.setFont(new Font(name:"Times new roman", Font.PLAIN, size:14));
49         return label;
50     });
51
BLACKBOX Chat Add Logs Code Agent Improve Code Java: Ready Share Code Link Ln 204, Col 61 Spaces: 4 UTF-8 CRLF () Java BLACKBOXAI: Open Chat Prettier
Type here to search 19°C عائم جزيتا ENG 7:55 PM 12/25/2024
```

```
File Edit Selection View Go Run ... library
J App.java J TablePanel.java X DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
22 private void initializeComponents() {
51
52     JScrollPane scrollPane = new JScrollPane(table);
53
54
55     JViewport viewport = scrollPane.getViewport();
56     viewport.setBackground(new Color(r:84, g:58, b:20));
57
58     scrollPane.setBackground(new Color(r:84, g:58, b:20));
59     scrollPane.getVerticalScrollBar().setBackground(new Color(r:240, g:187, b:120));
60     scrollPane.getHorizontalScrollBar().setBackground(new Color(r:240, g:187, b:120));
61
62     // Create button panel
63     JPanel buttonPanel = new JPanel();
64     JButton insertButton = new JButton(text:"Add");
65     JButton updateButton = new JButton(text:"Edit");
66     JButton deleteButton = new JButton(text:"Delete");
67     JButton refreshButton = new JButton(text:"Refresh");
68
69     styleButton(insertButton, new Color(r:250, g:223, b:161));
70     styleButton(updateButton, new Color(r:255, g:244, b:234));
71     styleButton(deleteButton, new Color(r:201, g:104, b:104));
72     styleButton(refreshButton, new Color(r:126, g:172, b:181));
73
74     buttonPanel.add(insertButton);
75     buttonPanel.add(updateButton);
76
BLACKBOX Chat Add Logs Code Agent Improve Code Java: Ready Share Code Link Ln 204, Col 61 Spaces: 4 UTF-8 CRLF () Java BLACKBOXAI: Open Chat Prettier
Type here to search 19°C عائم جزيتا ENG 7:55 PM 12/25/2024
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
22 private void initializeComponents() {
76 buttonPanel.add(deleteButton);
77 buttonPanel.add(refreshButton);
78 buttonPanel.setBackground(new Color(r:84, g:58, b:20));
79
80 add(buttonPanel, BorderLayout.NORTH);
81 add(scrollPane, BorderLayout.CENTER);
82
83 // Add listeners
84 insertButton.addActionListener(e -> showInsertDialog());
85 updateButton.addActionListener(e -> showUpdateDialog());
86 deleteButton.addActionListener(e -> deleteRecord());
87 refreshButton.addActionListener(e -> refreshTable());
88
89 // Initial load
90 refreshTable();
91 }
92
93 private void styleButton(JButton button, Color color) {
94 button.setBackground(color);
95 button.setForeground(Color.BLACK);
96 button.setFocusPainted(b:false);
97 button.setFont(new Font(name:"Times new roman", Font.BOLD, size:12));
98 button.setOpaque(isOpaque:true);
99 button.setContentAreaFilled(b:true);

```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
93 private void styleButton(JButton button, Color color) {
99 button.setContentAreaFilled(b:true);
100 button.setBorderPainted(b:false);
101 button.repaint();
102 }
103
104 private void refreshTable() {
105 try (Connection conn = DatabaseConnection.getConnection();
106 Statement stmt = conn.createStatement();
107 ResultSet rs = stmt.executeQuery("SELECT * FROM " + tableName)) {
108
109 DefaultTableModel model = new DefaultTableModel(displayNames, 0) {
110 @Override
111 public boolean isCellEditable(int row, int column) {
112 return false;
113 }
114 };
115
116 while (rs.next()) {
117 Object[] row = new Object[columnNames.length];
118 for (int i = 0; i < columnNames.length; i++) {
119 row[i] = rs.getObject(columnNames[i]);
120 }
121 model.addRow(row);
122 }
123 }

```



```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
104 private void refreshTable() {
124     table.setModel(model);
125
126 } catch (Exception ex) {
127     showError("Error loading data: " + ex.getMessage());
128 }
129 }
130
131 private void showInsertDialog() {
132     JDialog dialog = new JDialog((Frame) SwingUtilities.getWindowAncestor(this), "Add " + tableName, modal:true);
133     dialog.setLayout(new GridLayout(columnNames.length + 1, cols:2, hgap:5, vgap:5));
134
135     JTextField[] fields = new JTextField[columnNames.length];
136
137     // Create input fields
138     for (int i = 0; i < columnNames.length; i++) {
139         dialog.add(new JLabel(displayNames[i] + " :"));
140         fields[i] = new JTextField(columns:20);
141         fields[i].setBackground(new Color(r:247, g:238, b:211));
142         dialog.add(fields[i]);
143     }
144
145     // Add buttons
146     JButton saveButton = new JButton(text:"Save");
147     styleButton(saveButton, new Color(r:103, g:70, b:54));
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
131 private void showInsertDialog() {
147     styleButton(saveButton, new Color(r:103, g:70, b:54));
148     saveButton.addActionListener(e -> {
149         try {
150             insertRecord(fields);
151             dialog.dispose();
152             refreshTable();
153         } catch (Exception ex) {
154             showError("Error inserting record: " + ex.getMessage());
155         }
156     });
157
158     JButton cancelButton = new JButton(text:"Cancel");
159     styleButton(cancelButton, new Color(r:201, g:104, b:104));
160     cancelButton.addActionListener(e -> dialog.dispose());
161
162     dialog.add(saveButton);
163     dialog.add(cancelButton);
164
165     dialog.pack();
166     dialog.setLocationRelativeTo(this);
167     dialog.setVisible(b:true);
168 }
169
170 private void showUpdateDialog() {
171     int selectedRow = table.getSelectedRow();
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
170 private void showUpdateDialog() {
171     int selectedRow = table.getSelectedRow();
172     if (selectedRow == -1) {
173         showError(message: "Please select a Row to Edit.");
174         return;
175     }
176
177     JDialog dialog = new JDialog((Frame) SwingUtilities.getWindowAncestor(this), "Edit " + tableName, modal:true);
178     dialog.setLayout(new GridLayout(columnNames.length + 1, cols:2, hgap:5, vgap:5));
179
180     JTextField[] fields = new JTextField[columnNames.length];
181
182     // Create input fields with current values
183     for (int i = 0; i < columnNames.length; i++) {
184         dialog.add(new JLabel(displayNames[i] + ":"));
185         fields[i] = new JTextField(String.valueOf(table.getValueAt(selectedRow, i)), columns:20);
186         fields[i].setBackground(new Color(r:247, g:238, b:211));
187         dialog.add(fields[i]);
188     }
189
190     // Add buttons
191     JButton saveButton = new JButton(text:"Save");
192     styleButton(saveButton, new Color(r:103, g:70, b:54));
193     saveButton.addActionListener(e -> {
194         try {
195             updateRecord(fields);
196             dialog.dispose();
197             refreshTable();
198         } catch (Exception ex) {
199             showError("Error updating record: " + ex.getMessage());
200         }
201     });
202
203     JButton cancelButton = new JButton(text:"Cancel");
204     styleButton(cancelButton, new Color(r:201, g:104, b:104));
205     cancelButton.addActionListener(e -> dialog.dispose());
206
207     dialog.add(saveButton);
208     dialog.add(cancelButton);
209
210     dialog.pack();
211     dialog.setLocationRelativeTo(this);
212     dialog.setVisible(b:true);
213
214     private void insertRecord(JTextField[] fields) throws SQLException {
215         StringBuilder sql = new StringBuilder("INSERT INTO " + tableName + " (");
216         sql.append(String.join(delimiter:", ", columnNames));
217         sql.append(stmt:" VALUES (");
218         sql.append(String.join(delimiter:", ", fields));
219         sql.append(stmt:");");
220         String sqlStatement = sql.toString();
221         PreparedStatement pstmt = null;
222         try {
223             pstmt = conn.prepareStatement(sqlStatement);
224             pstmt.executeUpdate();
225         } catch (SQLException ex) {
226             showError("Error inserting record: " + ex.getMessage());
227         } finally {
228             if (pstmt != null) {
229                 pstmt.close();
230             }
231         }
232         refreshTable();
233     }
234
235     private void refreshTable() {
236         try {
237             stmt.executeUpdate("SELECT * FROM " + tableName);
238             table.setModel(DbUtils.resultSetToTableModel(stmt));
239         } catch (SQLException ex) {
240             showError("Error refreshing table: " + ex.getMessage());
241         }
242     }
243
244     private void showError(String message) {
245         JOptionPane.showMessageDialog(this, message, "Error", JOptionPane.ERROR_MESSAGE);
246     }
247
248     private void styleButton(JButton button, Color color) {
249         button.setBackground(color);
250         button.setForeground(Color.BLACK);
251     }
252 }
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
170 private void showUpdateDialog() {
194     try {
195         updateRecord(fields);
196         dialog.dispose();
197         refreshTable();
198     } catch (Exception ex) {
199         showError("Error updating record: " + ex.getMessage());
200     }
201 }
202
203 JButton cancelButton = new JButton(text:"Cancel");
204 styleButton(cancelButton, new Color(r:201, g:104, b:104));
205 cancelButton.addActionListener(e -> dialog.dispose());
206
207 dialog.add(saveButton);
208 dialog.add(cancelButton);
209
210 dialog.pack();
211 dialog.setLocationRelativeTo(this);
212 dialog.setVisible(b:true);
213
214 private void insertRecord(JTextField[] fields) throws SQLException {
215     StringBuilder sql = new StringBuilder("INSERT INTO " + tableName + " (");
216     sql.append(String.join(delimiter:", ", columnNames));
217     sql.append(stmt:" VALUES (");
218     sql.append(String.join(delimiter:", ", fields));
219     sql.append(stmt:");");
220     String sqlStatement = sql.toString();
221     PreparedStatement pstmt = null;
222     try {
223         pstmt = conn.prepareStatement(sqlStatement);
224         pstmt.executeUpdate();
225     } catch (SQLException ex) {
226         showError("Error inserting record: " + ex.getMessage());
227     } finally {
228         if (pstmt != null) {
229             pstmt.close();
230         }
231     }
232     refreshTable();
233 }
234
235 private void refreshTable() {
236     try {
237         stmt.executeUpdate("SELECT * FROM " + tableName);
238         table.setModel(DbUtils.resultSetToTableModel(stmt));
239     } catch (SQLException ex) {
240         showError("Error refreshing table: " + ex.getMessage());
241     }
242 }
243
244 private void showError(String message) {
245     JOptionPane.showMessageDialog(this, message, "Error", JOptionPane.ERROR_MESSAGE);
246 }
247
248 private void styleButton(JButton button, Color color) {
249     button.setBackground(color);
250     button.setForeground(Color.BLACK);
251 }
```



```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
215 private void insertRecord(JTextField[] fields) throws SQLException {
218     sql.append(str:" VALUES (");
219     sql.append("?.repeat(columnNames.length).replaceAll(regex:".?(=.)", replacement:"$0, ");
220     sql.append(str:")");
221
222     try (Connection conn = DatabaseConnection.getConnection();
223          PreparedStatement pstmt = conn.prepareStatement(sql.toString())) {
224
225         for (int i = 0; i < fields.length; i++) {
226             pstmt.setString(i + 1, fields[i].getText());
227         }
228
229         pstmt.executeUpdate();
230     } catch (Exception e) {
231         throw new RuntimeException(e);
232     }
233 }
234
235 private void updateRecord(JTextField[] fields) throws SQLException {
236     StringBuilder sql = new StringBuilder("UPDATE " + tableName + " SET ");
237     for (int i = 1; i < columnNames.length; i++) {
238         if (i > 1)
239             sql.append(str:", ");
240         sql.append(columnNames[i]).append(str:"=?");
241     }
242     sql.append(str:" WHERE ").append(columnNames[0]).append(str:"=?");
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
235 private void updateRecord(JTextField[] fields) throws SQLException {
243
244     try (Connection conn = DatabaseConnection.getConnection();
245          PreparedStatement pstmt = conn.prepareStatement(sql.toString())) {
246
247         // Set all fields except the ID
248         for (int i = 1; i < fields.length; i++) {
249             pstmt.setString(i, fields[i].getText());
250         }
251         // Set ID in WHERE clause
252         pstmt.setString(fields.length, fields[0].getText());
253
254         pstmt.executeUpdate();
255     } catch (Exception e) {
256         throw new RuntimeException(e);
257     }
258 }
259
260 private void deleteRecord() {
261     int selectedRow = table.getSelectedRow();
262     if (selectedRow == -1) {
263         showError(message:"Please select a record to delete");
264         return;
265     }
266 }
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999 }
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J TablePanel.java > TablePanel > showUpdateDialog()
7 public class TablePanel extends JPanel {
260 private void deleteRecord() {
266
267     if (JOptionPane.showConfirmDialog(this,
268         message:"Are you sure you want to delete this record?",
269         title:"Confirm Delete",
270         JOptionPane.YES_NO_OPTION) == JOptionPane.YES_OPTION) {
271
272         try {
273             String sql = "DELETE FROM " + tableName + " WHERE " + columnNames[0] + "=?";
274             try (Connection conn = DatabaseConnection.getConnection();
275                 PreparedStatement pstmt = conn.prepareStatement(sql)) {
276
277                 pstmt.setString(parameterIndex:1, String.valueOf(table.getValueAt(selectedRow, column:0)));
278                 pstmt.executeUpdate();
279                 refreshTable();
280             }
281         } catch (Exception ex) {
282             showError("Error deleting record: " + ex.getMessage());
283         }
284     }
285
286     private void showError(String message) {
287         JOptionPane.showMessageDialog(this, message, title:"Error", JOptionPane.ERROR_MESSAGE);
288     }
289 }
```

```
File Edit Selection View Go Run ... library
src > J App.java TablePanel.java DatabaseConnection.java
src > J DatabaseConnection.java > ...
4 public class DatabaseConnection {
5     private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=Library Database;encrypt=true;trustServerCertificate=true";
6     private static final String USER = "Abdo";
7     private static final String PASSWORD = "1324";
8
9     public static Connection getConnection() throws Exception {
10         Class.forName(className:"com.microsoft.sqlserver.jdbc.SQLServerDriver");
11         return DriverManager.getConnection(URL, USER, PASSWORD);
12     }
13 }
```