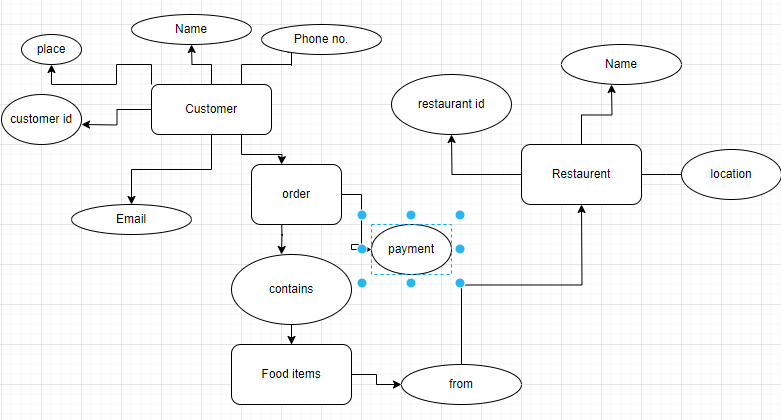
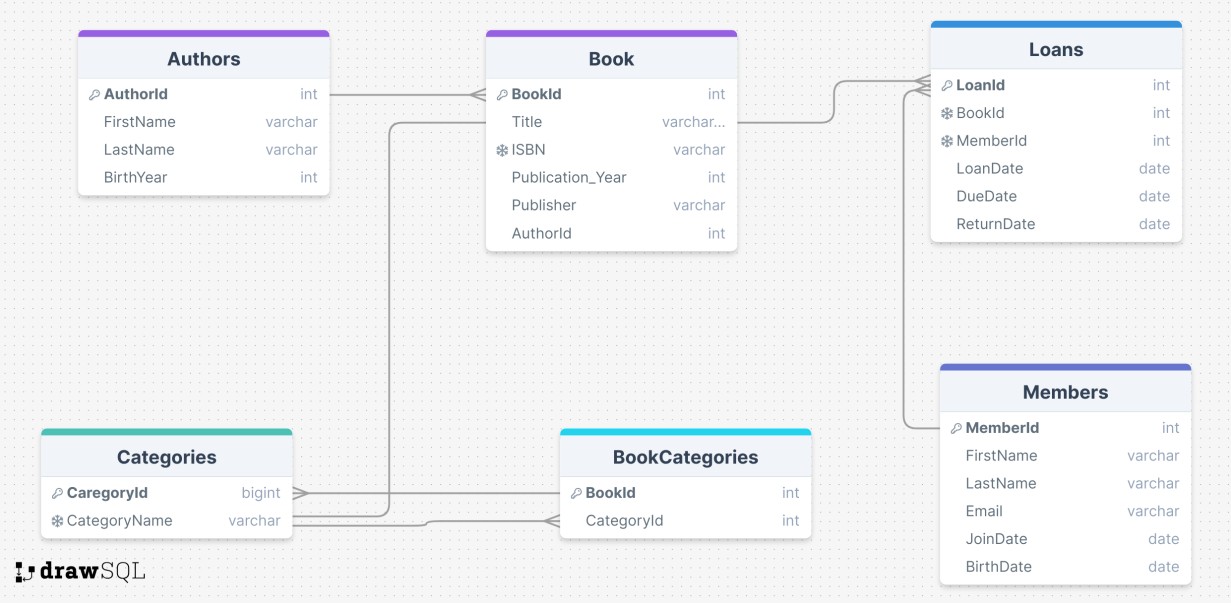
**Assignment 1:** Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.



**Assignment 2:** Design a database schema for a library system, including tables, fields, and constraints like NOT NULL, UNIQUE, and CHECK. Include primary and foreign keys to establish relationships between tables.



## Queries:

create database Library\_System; use Library\_System;

CREATE TABLE Authors (

AuthorID INT PRIMARY KEY AUTO\_INCREMENT, FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

BirthYear INT CHECK (BirthYear >= 0)

);

CREATE TABLE Books (

BookID INT PRIMARY KEY AUTO\_INCREMENT, Title VARCHAR(255) NOT NULL,

ISBN VARCHAR(13) UNIQUE NOT NULL,

PublicationYear INT CHECK (PublicationYear >= 0), Publisher VARCHAR(255),

AuthorID INT,

FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)

);

CREATE TABLE Members (

MemberID INT PRIMARY KEY AUTO\_INCREMENT, FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL, Email VARCHAR(255) UNIQUE NOT NULL, JoinDate DATE NOT NULL,

BirthDate DATE

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY AUTO\_INCREMENT, BookID INT NOT NULL,

MemberID INT NOT NULL, LoanDate DATE NOT NULL, DueDate DATE NOT NULL,

ReturnDate DATE,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (MemberID) REFERENCES Members(MemberID)

);

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT, CategoryName VARCHAR(255) UNIQUE NOT NULL

);

CREATE TABLE BookCategories ( BookID INT NOT NULL,

CategoryID INT NOT NULL, PRIMARY KEY (BookID, CategoryID),

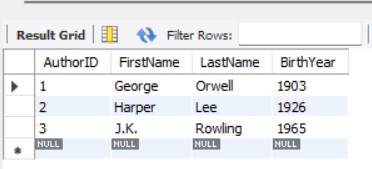
FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

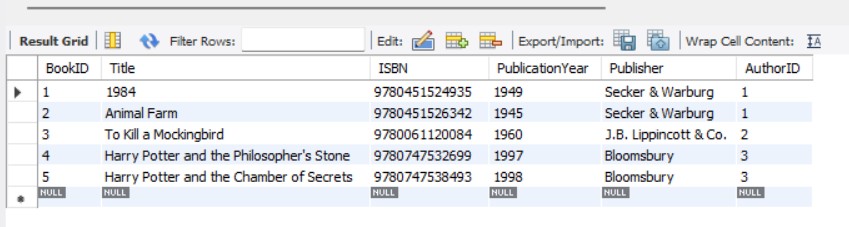
);

***OUTPUT:***

SELECT \* FROM library\_system.authors;



SELECT \* FROM library\_system.books;



**Assignment 3**: Explain the ACID properties of a transaction in your own words. Write SQL statements to simulate a transaction that includes locking and demonstrate different isolation levels to show concurrency control.

### ACID Properties of a Transaction

ACID stands for Atomicity, Consistency, Isolation, and Durability. These are the key properties that ensure reliable processing of database transactions.

1. **Atomicity**:

* Ensures that a transaction is treated as a single unit, which either completes entirely or does not happen at all. If any part of the transaction fails, the entire transaction fails and the database is left unchanged.

1. **Consistency**:

* Ensures that a transaction brings the database from one valid state to another, maintaining the database's integrity. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

1. **Isolation**:

* Ensures that concurrently executing transactions do not affect each other. The intermediate state of a transaction is invisible to other transactions, preventing issues like dirty reads, non-repeatable reads, and phantom reads.

1. **Durability**:

* Ensures that once a transaction is committed, it remains so, even in the event of a system crash. The changes made by the transaction are permanent and stored in non-volatile memory.

CREATE TABLE accounts (

id INT PRIMARY KEY,

balance DECIMAL(10, 2)

);

INSERT INTO accounts (id, balance) VALUES (1, 1000.00), (2, 2000.00);

START TRANSACTION;

SELECT balance FROM accounts WHERE id = 1 FOR UPDATE;

SELECT balance FROM accounts WHERE id = 2 FOR UPDATE;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

UPDATE accounts SET balance = balance + 100 WHERE id = 2;

COMMIT;

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

START TRANSACTION;

SELECT balance FROM accounts WHERE id = 1;

START TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

SELECT balance FROM accounts WHERE id = 1;

ROLLBACK;

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

START TRANSACTION;

SELECT balance FROM accounts WHERE id = 1;

START TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

COMMIT;

SELECT balance FROM accounts WHERE id = 1;

ROLLBACK;

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

SELECT balance FROM accounts WHERE id = 1;

START TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

COMMIT;

SELECT balance FROM accounts WHERE id = 1;

ROLLBACK;

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

START TRANSACTION;

SELECT balance FROM accounts WHERE id = 1;

START TRANSACTION;

UPDATE accounts SET balance = balance - 100 WHERE id = 1;

ROLLBACK;

SELECT balance FROM accounts WHERE id = 1;



**Assignment 4:** Write SQL statements to CREATE a new database and tables that reflect the library schema you designed earlier. Use ALTER statements to modify the table structures and DROP statements to remove a redundant table

## SQL Statement:

CREATE DATABASE LibraryDB;

USE LibraryDB;

CREATE TABLE Authors (

AuthorID INT PRIMARY KEY AUTO\_INCREMENT, FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

BirthYear INT CHECK (BirthYear >= 0)

);

CREATE TABLE Books (

BookID INT PRIMARY KEY AUTO\_INCREMENT, Title VARCHAR(255) NOT NULL,

ISBN VARCHAR(13) UNIQUE NOT NULL,

PublicationYear INT CHECK (PublicationYear >= 0), Publisher VARCHAR(255),

AuthorID INT,

FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)

);

CREATE TABLE Members (

MemberID INT PRIMARY KEY AUTO\_INCREMENT, FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL, Email VARCHAR(255) UNIQUE NOT NULL, JoinDate DATE NOT NULL,

BirthDate DATE

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY AUTO\_INCREMENT, BookID INT NOT NULL,

MemberID INT NOT NULL, LoanDate DATE NOT NULL, DueDate DATE NOT NULL,

ReturnDate DATE,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (MemberID) REFERENCES Members(MemberID)

);

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT, CategoryName VARCHAR(255) UNIQUE NOT NULL

);

CREATE TABLE BookCategories ( BookID INT NOT NULL,

CategoryID INT NOT NULL, PRIMARY KEY (BookID, CategoryID),

FOREIGN KEY (BookID) REFERENCES Books(BookID),

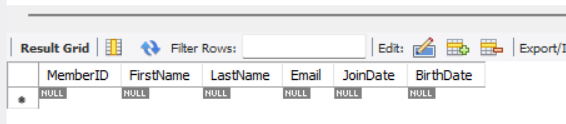
FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

**Author Table;**



**Member Table:**



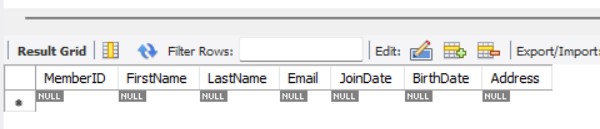
## ALTER Statements to Modify Table Structures:

-- Add MiddleName column to Authors ALTER TABLE Authors

ADD COLUMN MiddleName VARCHAR(255);

-- Add Address column to Members ALTER TABLE Members

ADD COLUMN Address VARCHAR(255);



## DROP Statement to Remove a Redundant Table

CREATE TABLE OldCategories (

OldCategoryID INT PRIMARY KEY AUTO\_INCREMENT, OldCategoryName VARCHAR(255) UNIQUE NOT NULL);

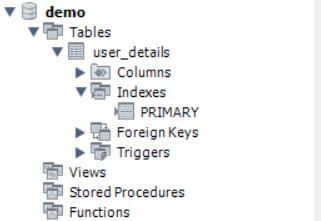


-- Drop the redundant table OldCategories DROP TABLE OldCategories;



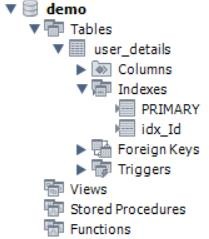
**Assignment 5:** Demonstrate the creation of an index on a table and discuss how it improves query performance. Use a DROP INDEX statement to remove the index and analyze the impact on query execution.

## Database structure without having Index:



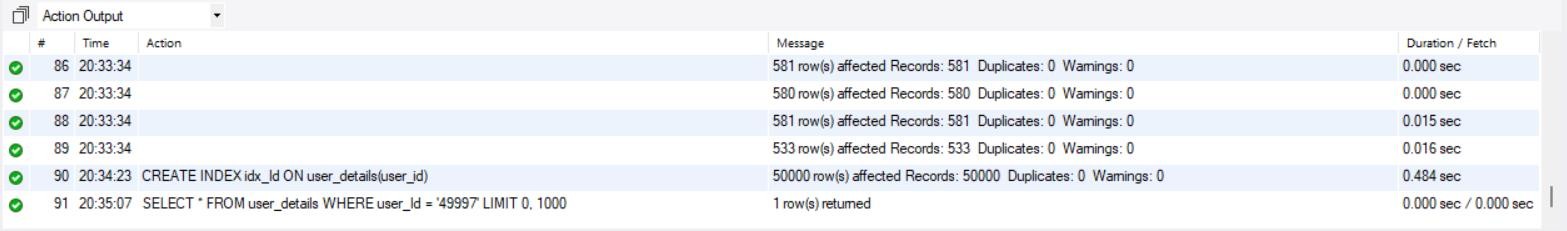
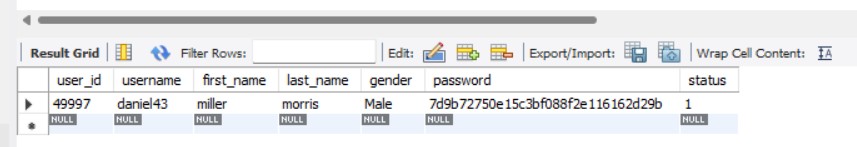
**Create an index on the user\_details(user\_id):**

CREATE INDEX idx\_Id ON user\_details(user\_id);



---- Query execution with index

SELECT \* FROM user\_details WHERE user\_Id = '49997';

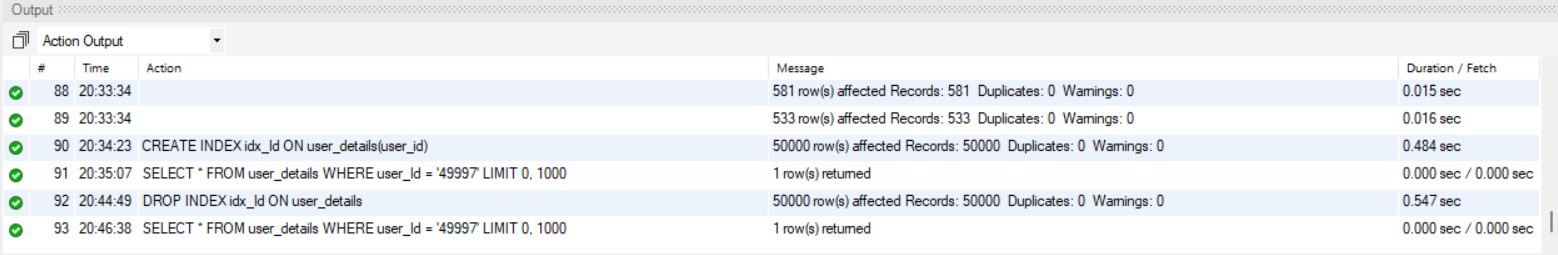
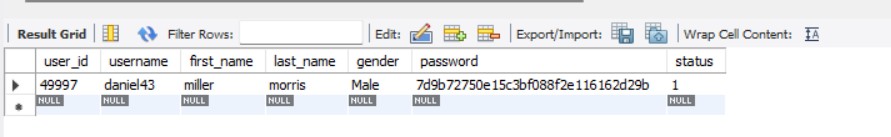


## Drop the index on the user\_details(user\_id):

DROP INDEX idx\_Id ON user\_details ;

-- Query execution without index

SELECT \* FROM user\_details WHERE user\_Id = '49997';



**Assignment 6:** Create a new database user with specific privileges using the CREATE USER and GRANT commands. Then, write a script to REVOKE certain privileges and DROP the user.

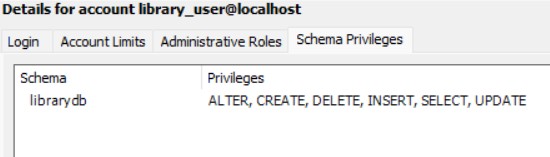
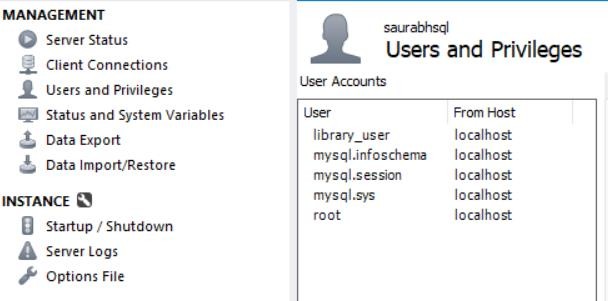
## Create a New Database User and grant permission:

-- Step 1: Create a new user

CREATE USER 'library\_user'@'localhost' IDENTIFIED BY 'pass123'; GRANT SELECT, INSERT, UPDATE, DELETE ON LibraryDB.\* TO

'library\_user'@'localhost';

GRANT CREATE, ALTER ON LibraryDB.\* TO 'library\_user'@'localhost';



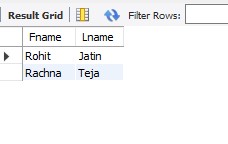
use accounts;

create table Shobhitfrd(Fname varchar(50),Lname varchar(50));

insert into Shobhitfrd values('Radha','Priya'),('Rohit','Jatin'),('Rachna','Teja');

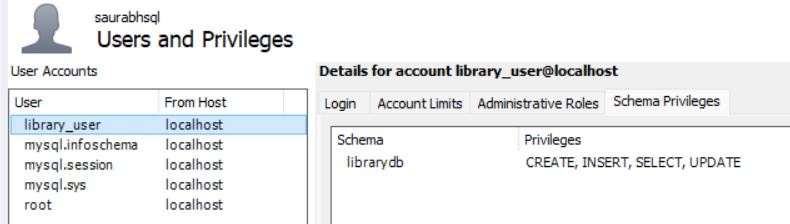
set sql\_safe\_updates=0;

delete from Shobhitfrd where Lname='Priya';



# Revoke DELETE AND ALETER privilege on all tables in LibraryDB:

REVOKE DELETE ON accounts.\* FROM 'library\_user'@'localhost'; REVOKE ALTER ON accounts.\* FROM 'library\_user'@'localhost';



# Drop the user:

DROP USER 'library\_user'@'localhost';

