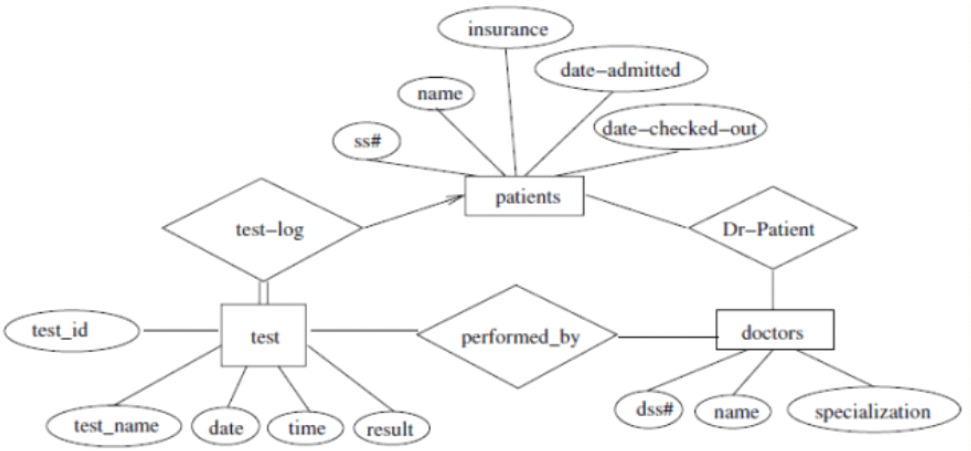
Assignment 1: Analyze a given business scenario and create an ER diagram that includes entities, relationships, attributes, and cardinality. Ensure that the diagram reflects proper normalization up to the third normal form.



E-R diagram for a hospital.

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.

```
CREATE TABLE Authors (
    AuthorID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    UNIQUE (FirstName, LastName)
);
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY AUTO_INCREMENT,
    CategoryName VARCHAR(100) NOT NULL UNIQUE
);
```

```
CREATE TABLE Books (
    BookID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255) NOT NULL,
    ISBN VARCHAR(13) NOT NULL UNIQUE,
    Publisher VARCHAR(100),
    PublishYear YEAR CHECK (PublishYear <= YEAR(CURDATE())),
    CategoryID INT,
    AuthorID INT,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);
CREATE TABLE Members (
    MemberID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15),
    JoinDate DATE NOT NULL,
    UNIQUE (FirstName, LastName, Email)
);
CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    LoanDate DATE NOT NULL,
```

```
    ReturnDate DATE,
    DueDate DATE NOT NULL,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    CHECK (ReturnDate IS NULL OR ReturnDate >= LoanDate),
    CHECK (DueDate >= LoanDate)
);
CREATE TABLE LibraryStaff (
    StaffID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15),
    HireDate DATE NOT NULL,
    Role VARCHAR(50) NOT NULL,
    CHECK (Role IN ('Librarian', 'Assistant', 'Manager'))
);
CREATE TABLE BookCopies (
    CopyID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT NOT NULL,
    ShelfLocation VARCHAR(100),
    Status VARCHAR(20) NOT NULL DEFAULT 'Available',
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    CHECK (Status IN ('Available', 'Loaned', 'Reserved', 'Lost'))
);
```

```
CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
    MemberID INT NOT NULL,
    BookID INT NOT NULL,
    ReservationDate DATE NOT NULL,
    Status VARCHAR(20) NOT NULL DEFAULT 'Pending',
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    CHECK (Status IN ('Pending', 'Fulfilled', 'Cancelled'))
);
```

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 is an acronym that stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure reliable processing of database transactions.

Atomicity: Ensures that a transaction is treated as a single unit, which either completely succeeds or completely fails. If any part of the transaction fails, the entire transaction is rolled back.

Consistency: Ensures that a transaction brings the database from one valid state to another, maintaining all predefined rules and constraints. This guarantees that any data written to the database must be valid according to all defined rules.

Isolation: Ensures that the operations of a transaction are invisible to other transactions until the transaction is complete. This prevents transactions from interfering with each other and ensures consistency when transactions are executed concurrently.

Durability: Ensures that once a transaction has been committed, it will remain so, even in the event of a system failure. This guarantees that the results of the transaction are permanently recorded in the database.

SQL Statements to Simulate a Transaction with Locking

```
CREATE TABLE Accounts (
    AccountID INT PRIMARY KEY,
    Balance DECIMAL(10, 2) NOT NULL
);

INSERT INTO Accounts (AccountID, Balance) VALUES (1, 1000.00), (2, 2000.00);
-- Start the transaction
START TRANSACTION;

-- Lock the rows to prevent other transactions from accessing them
SELECT Balance FROM Accounts WHERE AccountID = 1 FOR UPDATE;
SELECT Balance FROM Accounts WHERE AccountID = 2 FOR UPDATE;

-- Update the balances
UPDATE Accounts SET Balance = Balance - 100.00 WHERE AccountID = 1;
UPDATE Accounts SET Balance = Balance + 100.00 WHERE AccountID = 2;

-- Commit the transaction
COMMIT;
```

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.

```
-- Create the new database

CREATE DATABASE LibraryDB;


-- Use the new database

USE LibraryDB;


-- Create Authors table

CREATE TABLE Authors (

    AuthorID INT PRIMARY KEY AUTO_INCREMENT,

    FirstName VARCHAR(50) NOT NULL,

    LastName VARCHAR(50) NOT NULL,

    UNIQUE (FirstName, LastName)

);


-- Create Categories table

CREATE TABLE Categories (

    CategoryID INT PRIMARY KEY AUTO_INCREMENT,

    CategoryName VARCHAR(100) NOT NULL UNIQUE

);


-- Create Books table

CREATE TABLE Books (

    BookID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    Title VARCHAR(255) NOT NULL,
    ISBN VARCHAR(13) NOT NULL UNIQUE,
    Publisher VARCHAR(100),
    PublishYear YEAR CHECK (PublishYear <= YEAR(CURDATE())),
    CategoryID INT,
    AuthorID INT,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID),
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);

-- Create Members table
CREATE TABLE Members (
    MemberID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    MiddleName VARCHAR(50), -- Adding MiddleName column
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15),
    JoinDate DATE NOT NULL,
    UNIQUE (FirstName, LastName, Email)
);

-- Create Loans table
CREATE TABLE Loans (
    LoanID INT PRIMARY KEY AUTO_INCREMENT,
```

```
    BookID INT NOT NULL,
    MemberID INT NOT NULL,
    LoanDate DATE NOT NULL,
    ReturnDate DATE,
    DueDate DATE NOT NULL,
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    CHECK (ReturnDate IS NULL OR ReturnDate >= LoanDate),
    CHECK (DueDate >= LoanDate)
);

-- Create LibraryStaff table
CREATE TABLE LibraryStaff (
    StaffID INT PRIMARY KEY AUTO_INCREMENT,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    PhoneNumber VARCHAR(15),
    HireDate DATE NOT NULL,
    Role VARCHAR(50) NOT NULL,
    CHECK (Role IN ('Librarian', 'Assistant', 'Manager'))
);

-- Create BookCopies table
CREATE TABLE BookCopies (
```

```
    CopyID INT PRIMARY KEY AUTO_INCREMENT,
    BookID INT NOT NULL,
    ShelfLocation VARCHAR(100),
    Status VARCHAR(20) NOT NULL DEFAULT 'Available',
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    CHECK (Status IN ('Available', 'Loaned', 'Reserved', 'Lost'))
);

-- Create Reservations table
CREATE TABLE Reservations (
    ReservationID INT PRIMARY KEY AUTO_INCREMENT,
    MemberID INT NOT NULL,
    BookID INT NOT NULL,
    ReservationDate DATE NOT NULL,
    Status VARCHAR(20) NOT NULL DEFAULT 'Pending',
    FOREIGN KEY (MemberID) REFERENCES Members(MemberID),
    FOREIGN KEY (BookID) REFERENCES Books(BookID),
    CHECK (Status IN ('Pending', 'Fulfilled', 'Cancelled'))
);

-- Add Edition to Books table
ALTER TABLE Books
ADD COLUMN Edition VARCHAR(50);

-- Drop foreign key constraint from Books table
```

```
ALTER TABLE Books DROP FOREIGN KEY Books_ibfk_1;


-- Drop Categories table

DROP TABLE Categories;
```

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.

```
-- Use the LibraryDB database

USE LibraryDB;


-- Create the index on the Title column of the Books table

CREATE INDEX idx_title ON Books (Title);


-- Analyze query execution with the index

EXPLAIN SELECT * FROM Books WHERE Title = 'Some Book Title';


-- Run the query with the index

SELECT * FROM Books WHERE Title = 'Some Book Title';


-- Drop the index on the Title column of the Books table

DROP INDEX idx_title ON Books;


-- Analyze query execution without the index

EXPLAIN SELECT * FROM Books WHERE Title = 'Some Book Title';
```

-- Run the query without the index

SELECT * FROM Books WHERE Title = 'Some Book Title';

EXPLAIN Results Interpretation

With the Index: The EXPLAIN output will show that the database uses the idx_title index to find the matching rows. The type column will typically show 'ref' or 'index', indicating index usage.

Without the Index: The EXPLAIN output will show a full table scan. The type column will show 'ALL', indicating that the database scans all rows in the table.

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.

-- Use the MySQL root user or another user with appropriate privileges to run these commands

-- Step 1: Create the user 'library_user'

CREATE USER 'library_user'@'localhost' IDENTIFIED BY 'SecurePassword123';

-- Step 2: Grant SELECT, INSERT, UPDATE, and DELETE privileges on LibraryDB.* to 'library_user'

GRANT SELECT, INSERT, UPDATE, DELETE ON LibraryDB.* TO 'library_user'@'localhost';

-- Step 3: Revoke DELETE privilege from 'library_user'

REVOKE DELETE ON LibraryDB.* FROM 'library_user'@'localhost';

-- Step 4: Drop the user 'library_user'

DROP USER 'library_user'@'localhost';

Assignment 7: Prepare a series of SQL statements to INSERT new records into the library tables, UPDATE existing records with new information, and DELETE records based on specific criteria. Include BULK

INSERT operations to load data from an external source.

```
-- Use the LibraryDB database

USE LibraryDB;


-- Insert records into the Authors table

INSERT INTO Authors (FirstName, LastName) VALUES ('George', 'Orwell'), ('Jane', 'Austen'), ('Mark', 'Twain');


-- Insert records into the Categories table

INSERT INTO Categories (CategoryName) VALUES ('Fiction'), ('Non-Fiction'), ('Science'), ('Biography');


-- Insert records into the Books table

INSERT INTO Books (Title, ISBN, Publisher, PublishYear, CategoryID, AuthorID)

VALUES

('1984', '9780451524935', 'Signet Classics', 1949, 1, 1),

('Pride and Prejudice', '9781503290563', 'CreateSpace Independent Publishing', 1813, 1, 2),

('Adventures of Huckleberry Finn', '9780486280615', 'Dover Publications', 1884, 1, 3);


-- Insert records into the Members table

INSERT INTO Members (FirstName, LastName, Email, PhoneNumber, JoinDate)

VALUES

('Alice', 'Johnson', 'alice.johnson@example.com', '555-1234', '2023-01-10'),

('Bob', 'Smith', 'bob.smith@example.com', '555-5678', '2023-01-15');


-- Insert records into the LibraryStaff table

INSERT INTO LibraryStaff (FirstName, LastName, Email, PhoneNumber, HireDate, Role)
```

```
VALUES
('Emily', 'Brown', 'emily.brown@example.com', '555-8765', '2022-02-01', 'Librarian'),
('David', 'Wilson', 'david.wilson@example.com', '555-4321', '2022-03-01', 'Assistant');

-- Insert records into the BookCopies table
INSERT INTO BookCopies (BookID, ShelfLocation, Status)
VALUES
(1, 'Shelf A1', 'Available'),
(2, 'Shelf B1', 'Available'),
(3, 'Shelf C1', 'Available');

-- Insert records into the Reservations table
INSERT INTO Reservations (MemberID, BookID, ReservationDate, Status)
VALUES
(1, 1, '2023-01-20', 'Pending'),
(2, 2, '2023-01-21', 'Pending');

-- Update the email address of a member
UPDATE Members
SET Email = 'alice.j.newemail@example.com'
WHERE MemberID = 1;

-- Update the status of a book copy
UPDATE BookCopies
SET Status = 'Loaned'
```

```
WHERE CopyID = 1;


-- Update the due date of a loan
UPDATE Loans
SET DueDate = '2023-02-28'
WHERE LoanID = 1;


-- Delete a member who has not borrowed any books and joined over a year ago
DELETE FROM Members
WHERE MemberID = 2
AND MemberID NOT IN (SELECT MemberID FROM Loans)
AND JoinDate < DATE_SUB(CURDATE(), INTERVAL 1 YEAR);


-- Delete all reservations that are in 'Pending' status for more than a month
DELETE FROM Reservations
WHERE Status = 'Pending'
AND ReservationDate < DATE_SUB(CURDATE(), INTERVAL 1 MONTH);


-- Bulk insert data from a CSV file
LOAD DATA INFILE '/path/to/your/books.csv'
INTO TABLE Books
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
```

(Title, ISBN, Publisher, PublishYear, CategoryID, AuthorID);