**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.**

->Here is a proposed database schema for a library system:

**Tables**:

1.**Authors**:

AuthorID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each author

FirstName (varchar, NOT NULL): author's first name

LastName (varchar, NOT NULL): author's last name

BirthDate (date): author's birth date

**2.Books**

BookID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each book

Title (varchar, NOT NULL): book title

ISBN (varchar, NOT NULL, UNIQUE): book ISBN

PublicationDate (date): book publication date

AuthorID (integer, NOT NULL): foreign key referencing the Authors table

PublisherID (integer, NOT NULL): foreign key referencing the Publishers table

**3.Publishers**

PublisherID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each publisher

Name (varchar, NOT NULL): publisher name

Address (varchar): publisher address

Phone (varchar): publisher phone number

**4.Members**

MemberID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each member

FirstName (varchar, NOT NULL): member's first name

LastName (varchar, NOT NULL): member's last name

Email (varchar, NOT NULL, UNIQUE): member's email address

Phone (varchar): member's phone number

Address (varchar): member's address

**5.Loans**

LoanID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each loan

BookID (integer, NOT NULL): foreign key referencing the Books table

MemberID (integer, NOT NULL): foreign key referencing the Members table

LoanDate (date, NOT NULL): loan date

DueDate (date, NOT NULL): due date

ReturnDate (date): return date (optional)

**6.Fines**

FineID (primary key, integer, NOT NULL, UNIQUE): unique identifier for each fine

LoanID (integer, NOT NULL): foreign key referencing the Loans table

FineAmount (decimal, NOT NULL): fine amount

Paid (boolean, NOT NULL, default: false): indicates whether the fine has been paid

**Constraints:**

Authors: BirthDate must be in the past

Books: PublicationDate must be in the past

Loans: LoanDate must be in the past, DueDate must be in the future, and ReturnDate must be in the past if not null

Fines: FineAmount must be greater than 0

**Relationships:**

An author can write many books (one-to-many): Books.AuthorID references Authors.AuthorID

A book is written by one author (many-to-one): Books.AuthorID references Authors.AuthorID

A publisher publishes many books (one-to-many): Books.PublisherID references Publishers.PublisherID

A member can borrow many books (one-to-many): Loans.MemberID references Members.MemberID

A book can be borrowed by many members (many-to-many): Loans.BookID references Books.BookID

A loan can have one fine (one-to-one): Fines.LoanID references Loans.LoanID

This schema captures the essential information for a library system, including authors, books, publishers, members, loans, and fines. The relationships between tables are established using primary and foreign keys, ensuring data consistency and integrity.

**Write a SELECT query to retrieve all columns from a 'customers' table, and modify it to return only the customer name and email address for customers in a specific city.**

To retrieve all columns from the 'customers' table:

**SELECT \* FROM customers;**

Retrieve only customer name and email address for customers in a specific city:

**SELECT customer\_name, email\_address**

**FROM customers**

**WHERE city = 'Hyderabad';**

**Craft a query using an INNER JOIN to combine 'orders' and 'customers' tables for customers in a specified region, and a LEFT JOIN to display all customers including those without orders.**

->INNER JOIN to combine ‘orders’ and ‘customers’ tables for customers in a specified region:

**SELECT customers.customer\_id, customers.name, orders.order\_id, orders.order\_date**

**FROM customers**

**INNER JOIN orders ON customers.customer\_id = orders.customer\_id**

**WHERE customers.region = 'South';**

->This query will return all columns from both customers and orders tables, but only for customers who are in the 'North' region and have at least one order.

->LEFT JOIN to display all customers including those without orders:

**SELECT customers.customer\_id, customers.name, orders.order\_id, orders.order\_date**

**FROM customers**

**LEFT JOIN orders ON customers.customer\_id = orders.customer\_id**

**WHERE customers.region = 'North';**

->This query will return all columns from both customers and orders tables, including customers who are in the 'North' region but do not have any orders. The orders columns will be NULL for customers without orders.