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

Example Business Scenario

Scenario: Online Bookstore

An online bookstore sells books to customers. Each book has a unique ISBN, title, author(s), genre, and price. Customers have a unique customer ID, name, email, and shipping address. Customers can place orders, and each order can contain multiple books. An order has a unique order ID, order date, and total amount. The bookstore also keeps track of the stock levels for each book.

Steps to Create the ER Diagram

*1. **Identify Entities and Attributes***

*- **Book**: ISBN (Primary Key), Title, Price, Genre*

*- **Author**: AuthorID (Primary Key), Name, Biography*

*- **Customer**: CustomerID (Primary Key), Name, Email, ShippingAddress*

*- **Order**: OrderID (Primary Key), OrderDate, TotalAmount*

*- **OrderDetail**: OrderDetailID (Primary Key), OrderID (Foreign Key), ISBN (Foreign Key), Quantity*

*- **Stock**: StockID (Primary Key), ISBN (Foreign Key), QuantityInStock*

*2. **Identify Relationships***

*- **Book-Author**: Many-to-Many (since a book can have multiple authors and an author can write multiple books)*

*- **Customer-Order**: One-to-Many (since a customer can place multiple orders)*

*- **Order-OrderDetail**: One-to-Many (since an order can contain multiple books)*

*- **Book-OrderDetail**: Many-to-Many (since a book can appear in multiple orders and an order can contain multiple books)*

*- **Book-Stock**: One-to-One (since each book has a unique stock record)*

*3. **Normalize to 3NF***

- Ensure no transitive dependencies (attributes depending on non-key attributes)

- Separate many-to-many relationships into junction tables

ER Diagram

Let's represent this in an ER diagram.

```plaintext
+------------------+          +---------------+        +------------------+
|     Customer     |          |     Order     |        |      Book        |
+------------------+          +---------------+        +------------------+
| CustomerID (PK)  |<----------o| OrderID (PK)  |          | ISBN (PK)        |
| Name             |          | OrderDate     |    o------| Title            |
| Email            |          | TotalAmount   |    |     | Price            |
| ShippingAddress  |          +---------------+    |     | Genre            |
+------------------+                          |     +------------------+
                                              |
+------------------+                          |
|   OrderDetail    |<---------------------------------o
+------------------+
| OrderDetailID (PK)|
| OrderID (FK)      |
| ISBN (FK)         |
| Quantity          |
+------------------+
```

```
+------------------+          +------------------+
|     Author     |          |     Stock      |
+------------------+          +------------------+
| AuthorID (PK)    |o---------- | StockID (PK)     |
| Name           |           | ISBN (FK)      |
| Biography      |           | QuantityInStock |
+------------------+          +------------------+


+------------------+
|  BookAuthor     |
+------------------+
| BookAuthorID (PK)|
| ISBN (FK)       |
| AuthorID (FK)    |
+------------------+
```

Description

1. **Customer to Order**:
   - One-to-Many: A customer can place multiple orders, but each order is placed by one customer.
   - CustomerID is the primary key in the Customer table and a foreign key in the Order table.

2. **Order to OrderDetail**:
   - One-to-Many: Each order can have multiple order details, but each order detail is for one order.

- OrderID is the primary key in the Order table and a foreign key in the OrderDetail table.

*3. **Book to OrderDetail**:*

- Many-to-Many: Each book can appear in multiple orders, and each order can contain multiple books.

- Handled by the OrderDetail junction table.

*4. **Book to Author**:*

- Many-to-Many: Each book can have multiple authors, and each author can write multiple books.

- Handled by the BookAuthor junction table.

*5. **Book to Stock**:*

- One-to-One: Each book has a unique stock record.

- ISBN is the primary key in the Book table and a foreign key in the Stock table.

*6. **Normalization**:*

- Each entity is in 3NF as all attributes depend only on the primary key, and there are no transitive dependencies.

This ER diagram ensures the database is well-structured, normalized, and free from anomalies, making it efficient for operations such as querying, updating, and maintaining data integrity.

Assignment 1: 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.

*SELECT * FROM customers;*

SELECT customer_name, email

FROM customers

WHERE city = 'hyderabad';

Assignment 2: 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 is to combine orders and customers tables for customers in a specified region:

SELECT customers.customer_id, customers.customer_name, customers.email, orders.order_id, orders.order_date

FROM customers

INNER JOIN orders ON customers.customer_id = orders.customer_id

WHERE customers.region = 'specified_region';

Replace specified region with actual region you are intrested in:

SELECT customers.customer_id, customers.customer_name, customers.email, orders.order_id, orders.order_date

FROM customers

LEFT JOIN orders ON customers.customer_id = orders.customer_id;

Assignment 3: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.

*1.**Author's table**:*

field	***Data type***	***constraints***
author_id	*INT*	*PRIMARY KEY, AUTO_INCREMENT*
First_name	*VARCHAR(50)*	*NOT NULL*

Last_name	*VARCHAR(50)*	*NOT NULL*
dob	*DATE*	

*2.**books table***

field	*Data type*	*CONSTRAINTS*
book_id	*INT*	*PRIMARY KEY,AUTO_INCREMENT*
title	*VARCHAR(100)*	*NOT NULL*
isbn	*VARCHAR(13)*	*NOT NULL,UNIQUE*
Published year	*YEAR*	*CHECK(published_year>0)*
Author_id	*INT*	*FOREIGN KEY REFERENCES Authors(author_id)*

*3. **members table***

field	*Data type*	*constraints*
member_id	*INT*	*PRIMARY KEY,AUTO_INCREMENT*
first_name	*VARCHAR(50)*	*NOT NULL*
Last_name	*VARCHAR(50)*	*NOT NULL*
email	*VARCHAR(100)*	*NOT NULL, UNIQUE*
phone	*VARCHAR(15)*	

*4**.Loans table***

field	*Data type*	*constraints*
loan_id	*INT*	*PRIMARY KEY AUTO_INCREMENT*
book_id	*INT*	*NOT NULL,FOREIGN KEY REFERENCES Books(book_id)*
member_id	*INT*	*NOT NULL, FOREIGN KEY REFERENCES members(member_id)*
loan_date	*DATE*	*NOT NULL*
return_date	*DATE*	
due_date	*DATE*	*NOT NULL*

*5.**categories table***

field	***Data type***	***constraints***
category_id	***INT***	***PRIMARY KEY,AUTO_INCREMENT***
category_name	***VARCHAR(50)***	***NOT NULL, UNIQUE***

*6.**Book categories table***

field	*Data type*	*constraints*
book_id	*INT*	*NOT NULL,FOREIGN KEY REFERENCES Books(book_id)*
Category_id	*INT*	*NOT NULL, FOREIGN KEY REFERENCES Categories(category_id)*
PRIMARY KEY(book_id,category_id)		

- *7.**publishers table***

field	*Data type*	*constraints*
Publisher_id	*INT*	*PRIMARY KEY, AUTO _INCREMENT*
Publisher_name	*VARCHAR(100)*	*NOT NULL,UNIQUE*
Address	*VARCHAR(200)*	
phone	*VARCHAR(15)*	

*8.**Book publishers table***

field	*data type*	*constraints*
Book_id	*INT*	*NOT NULL,FOREIGN KEY REFERENCES Books(book_id)*
publisher_id	*INT*	*NOT NULL,FOREIGN KEY REFERENCES Publishers(publisher_id)*
PRIMARY KEY(book_id,publisher_id)		

Constraints explanation

- *Primary key:* uniquely identifies each record in a table.
- *Foreign key: establishes a link between records in two tables.*
- *Not null: ensures that a column cannot have a null value.*
- *Unique: ensures that all the values in a column are different.*
- *Check: ensures that all values in a column satisfy a specific condition.*

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 DATABASE

Create database:

```
Create database   LibrarySystem;

USE LibrarySystem;
```

Create tables:

```
CREATE TABLE Authors (
    author_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    dob DATE
);
```

Books table:

```
CREATE TABLE Books (
    book_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    isbn VARCHAR(13) NOT NULL UNIQUE,
    published_year YEAR CHECK (published_year > 0),
    author_id INT,
    FOREIGN KEY (author_id) REFERENCES Authors(author_id)
);
```

Members table:

```
CREATE TABLE Members (
```

```
    member_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(15)
);
```

Loans table:

```
CREATE TABLE Loans (
    loan_id INT AUTO_INCREMENT PRIMARY KEY,
    book_id INT NOT NULL,
    member_id INT NOT NULL,
    loan_date DATE NOT NULL,
    return_date DATE,
    due_date DATE NOT NULL,
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (member_id) REFERENCES Members(member_id)
);
```

Categories table:

```
CREATE TABLE Categories (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    category_name VARCHAR(50) NOT NULL UNIQUE
);
```

Book categories table:

```
CREATE TABLE BookCategories (
```

```
    book_id INT NOT NULL,
    category_id INT NOT NULL,
    PRIMARY KEY (book_id, category_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (category_id) REFERENCES Categories(category_id)
);
```

Publishers table:

```
CREATE TABLE Publishers (
    publisher_id INT AUTO_INCREMENT PRIMARY KEY,
    publisher_name VARCHAR(100) NOT NULL UNIQUE,
    address VARCHAR(200),
    phone VARCHAR(15)
);
```

Book publishers table:

```
CREATE TABLE BookPublishers (
    book_id INT NOT NULL,
    publisher_id INT NOT NULL,
    PRIMARY KEY (book_id, publisher_id),
    FOREIGN KEY (book_id) REFERENCES Books(book_id),
    FOREIGN KEY (publisher_id) REFERENCES Publishers(publisher_id)
);
```

Alter statements:

```
Alter table  Books
```

Add column language VARCHAR(50) DEFAULT 'English';

ALTER TABLE Members

MODIFY COLUMN phone VARCHAR(20);

Drop statements:

DROP TABLE BookCategories;