Step 2:

Specifications for the library database:

1. Managing Library Items
    - Each item should have a unique ID
    - Some items have extra details like a URL or a duration; every item should have sufficient attributes in either the default Items table, or a subclass.
    - Items that are part of a subclass should be inserted in both the items table and it's subclass table
    - Items are either currently "available", "borrowed", or marked as a "futureItem" meaning it might be added to the library later
2. Borrowing and Returning Items
    - Users can borrow items from the library and must return them by a due date
    - If a user does not return an item by its due date, then a fine should be calculated and be tied to the user's account
    - The system should prevent multiple users from borrowing the same item at the same time unless it is an online item.
    - Fines are marked as paid or unpaid
3. User Management
    - User's preference should be stored so it is easy to recommend events to the user
    - Users should be able to see their due dates, fines, and recommended events
4. Library Events and Recommendations
    - Each event is held in a room with a capacity limit.
    - Users should be able to attend events for free as long as there is enough space (capacity)
    - The system should recommend events to users based on their preferences
5. Library Staff and Management
    - The library has staff members that are responsible for managing specific events
    - Staff members have ID,s names, positions, and salaries
    - Certain staff should be able to manage loans and fines
6. General System requirements
    - The database should support multiple users borrowing and returning items at the same time
    - The database should allow the addition of new items, users, and events without corrupting the existing data
    - When an event is added, automatically insert the emails of users who prefer the event into a new table. Assume an external script is used to send all the recommendation emails

Entities and Attributes:

Entities	**Attributes**
Item	<u>itemID</u>, title, type, author, status (available, borrowed, futureItem, online)
CD (is an Item)	Duration (length of the media)

OnlineItem (is an Item)	Url (link to the online resource)
User	userID, preference (used to recommend users to events), email
Loan	loanID, due (date for when an item must be returned)
Fine	Amount, status (paid, unpaid)
Event	eventID, type (Book club, art show, etc.), date
Room	roomID, capacity
Personnel	staffID, name, position, salary
HelpRequests	requestID, email, message

Relationships:

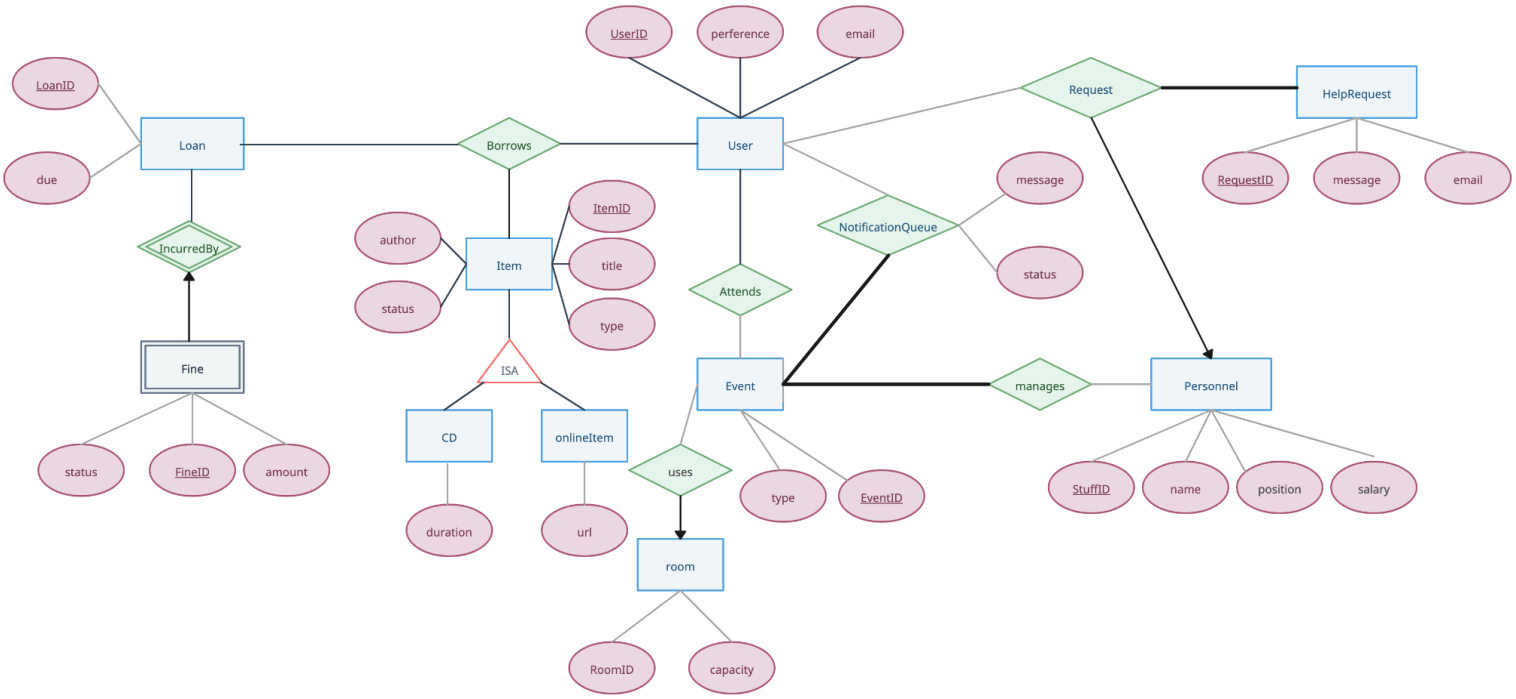
1. Borrows (User - Loan - Item)
    - User's can borrow multiple items and have multiple fines
    - An item can be borrowed by one user at a time
2. IncurredBy (Loan - Fine, many-to-one)
    - A fine is incurred by a specific loan
3. Attends (User - Event, many-to-many)
    - Users can attend multiple events
    - Events are free to attend
4. Uses (Event - Room, many-to-one)
    - Each event is held in a specific room
5. Manages (Personnel - Event, many-to-many)
    - Each event is managed by one or more staff
6. Request(User - Personnel - HelpRequest)
    - Each staff is received one or more user request
7. NotificationQueue (Event - User)
    - Each event will notice one or more user

Triggers:

1. Update item status on borrow (after insert on loan)
2. Update item status on return (after delete on loan)
3. Prevent item borrow if not available (before insert on loan)
4. Prevent CD insert to table CD if it is not existing in Item table
5. Prevent Online Item insert to table OnlineItem if it is not existing in Item table

Other:

1. A fine should be calculated and linked to the user automatically when a late item is returned
2. An external script should be used to send all emails in the notificationQueue
3. Staff can view user requests (stored in HelpRequests table) and are expected to directly email the user



Step 4:

All my FDs:
ItemID -> title, type, author, status
ItemID -> duration
ItemID -> url
UserID -> preference,email
LoanID -> due
FineID -> amount, status
EventID -> type
RoomID -> capacity
staffID -> position, name, salary
RequestID ->email, message
UserID, EventID ->message, status

The left side of the FDs in all of our relationships are superkeys, so there are no bad FDs and no partial dependencies. All non-primary key attributes are directly dependent on the primary key, so there are no transitive dependencies.