Project Specifications

1. **Items**:
    - Represents the various items available in the library or rental system.
    - Fields:
        - **ItemID**: A unique identifier for each item.
        - **Type**: The type or category of the item (could be a book, CD, DVD, etc.).
        - **Title**: The title of the item.
        - **AuthorArtist**: The author of a book or the artist of a CD/DVD.
        - **PublicationYear**: The year when the item was published or released.
        - **Genre**: Genre of the item (like fiction, non-fiction, rock, classical, etc.).
        - **Availability**: Status to denote if the item is available for borrowing.
        - **Stock**: The number of copies of the item available.
2. **Customers**:
    - Represents the library or system's customers.
    - Fields:
        - **CustomerID**: A unique identifier for each customer.
        - **Name**: Name of the customer.
        - **NumberOfLoans**: Total number of items a customer has borrowed.
        - **Balance**: Could be the balance of any unpaid dues or charges.
3. **Loans**:
    - Represents the loans or borrowings of items by customers.
    - Fields:
        - **LoanID**: Unique identifier for each loan transaction.

- **CustomerID**: The ID of the customer borrowing the item.
- **ItemID**: The ID of the item being borrowed.
- **LoanDate**: Date when the item was borrowed.
- **DueDate**: Date when the item is expected to be returned.
- **ReturnDate**: Actual date when the item was returned.
- Fee for late return is calculated based on the number of months the book is late for and deducted from Customer's balance.

4. **Employees**:
    - Represents the employees of the library or rental system.
    - Fields:
        - **EmployeeID**: A unique identifier for each employee.
        - **Name**: Name of the employee.
        - **Type**: Type of employment, which can either be "Paid" or "Volunteering".
5. **Events**:
    - Represents events organized by the library or system.
    - Fields:
        - **EventID**: A unique identifier for each event.
        - **EmployeeID**: The ID of the employee organizing or responsible for the event.
        - **RoomNumber**: Room or location where the event will be held.
        - **Name**: Name or title of the event.
        - **Audience**: Target audience for the event.
        - **Price**: Cost of attending the event.
        - The cost of the event is deducted from customer's balance.
        - **Number_of_Attendees**: Total number of attendees for the event.
        - **MAX_Attendees**: Maximum allowed number of attendees for the event.

6. **FutureItems**:
    - Represents items that the library or system plans to acquire in the future.
    - Fields:
        - **F_ItemID**: A unique identifier for each future item.
        - Most other fields are similar to the "Items" table, with the addition of **AvailabilityDate**, which denotes when the item will be available.
7. **Queries**:
    - Represents queries or concerns raised by customers to the staff.
    - Fields:
        - **QueryID**: A unique identifier for each query.
        - **CustomerID**: The ID of the customer raising the query.
        - **QueryText**: The text or description of the query.
        - **EmployeeID**: The ID of the employee addressing the query.
        - **Status**: The status of the query, either "Resolved" or "Unresolved".

A library database has specific needs and requirements to ensure smooth operations, cataloging, and user satisfaction. Let's break down the essential needs and requirements for a library database in simple terms:

1. **Cataloging Items**:
    - **Need**: Keep track of all items (books, magazines, DVDs, etc.).
    - **Simple Specification**: A list of all books and things with details like title, author, and publication year.
2. **Tracking Availability**:
    - **Need**: Know which items are available or checked out.
    - **Simple Specification**: A status for each item like "available" and "checked out".

3. **Managing Members**:
    - **Need**: Register and manage library members.
    - **Simple Specification**: A list of all people using the library with all needed information about them.
4. **Loan Management**:
    - **Need**: Monitor items borrowed by members, when they're due back, and any late fees.
    - **Simple Specification**: A record of which person took which book, when they took it, and when they should bring it back.
5. **Search Capability**:
    - **Need**: Help members and staff find items quickly.
    - **Simple Specification**: A search box to type in and find books or items by their title, author, or other details.
6. **Events and Classes**:
    - **Need**: Organize and schedule events or classes at the library.
    - **Simple Specification**: A calendar showing different events happening in the library.
7. **Employee Management**:
    - **Need**: Keep track of library staff and information about them.
    - **Simple Specification**: A list of all library workers
8. **Member Queries and Feedback**:
    - **Need**: Allow members to ask questions.
    - **Simple Specification**: A form for members to write questions and receive answers from library staff.
9. **Future Acquisitions**:
    - **Need**: Plan and keep track of items the library wants to buy or add in the future.
    - **Simple Specification**: A wishlist of books or items the library plans to get.

10. **Fine and Payment Management**:
    - **Need**: Manage any fines for late returns and collect payments.
    - **Simple Specification**: A record of any money a member owes for late returns and a way for them to pay.

Entity-relationship model and diagram

For the given database, I'll outline the main entities and their relationships.

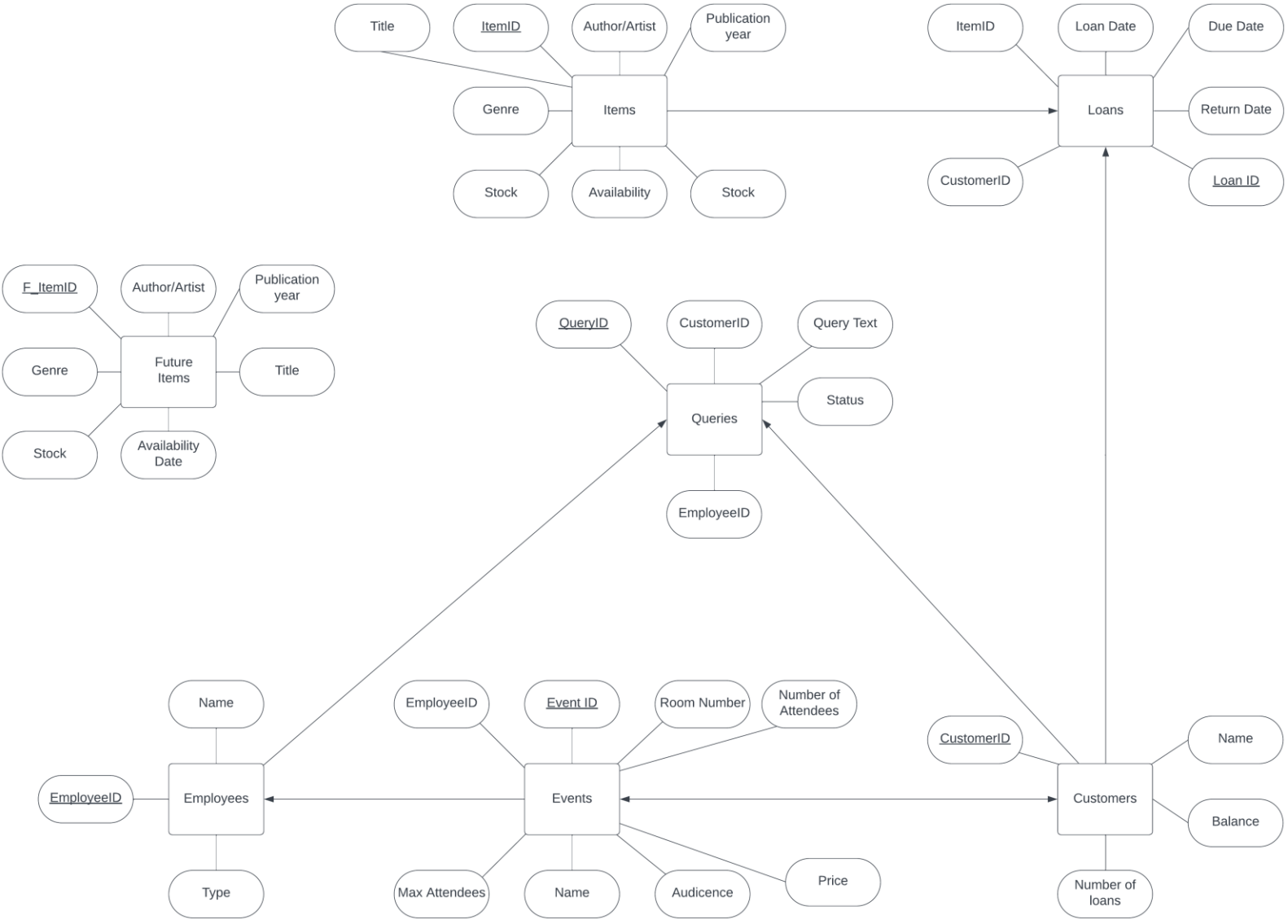
Entities:

1. **Items**
    - Attributes: ItemID, Type, Title, AuthorArtist, PublicationYear, Genre, Availability, Stock
2. **Customers**
    - Attributes: CustomerID, Name, NumberOfLoans, Balance
3. **Loans**
    - Attributes: LoanID, CustomerID, ItemID, LoanDate, DueDate, ReturnDate
4. **Employees**
    - Attributes: EmployeeID, Name, Type
5. **Events**
    - Attributes: EventID, EmployeeID, RoomNumber, Name, Audience, Price, Number_of_Attendees, MAX_Attendees
6. **FutureItems**
    - Attributes: F_ItemID, Type, Title, AuthorArtist, PublicationYear, Genre, AvailabilityDate, Stock
7. **Queries**
    - Attributes: QueryID, CustomerID, QueryText, EmployeeID, Status

Relationships:

1. **Loans-Customers**: One-to-Many from Customers to Loans.
    - A customer can have many loans, but each loan is associated with one customer.
    - Foreign Key: **CustomerID** in Loans references **CustomerID** in Customers.

2. **Loans-Items**: One-to-Many from Items to Loans.
    - An item can be part of many loans (over time), but each loan refers to one item.
    - Foreign Key: **ItemID** in Loans references **ItemID** in Items.
3. **Events-Employees**: Many-to-One from Events to Employees.
    - Many events can be organized or overseen by one employee, but each event is associated with one employee.
    - Foreign Key: **EmployeeID** in Events references **EmployeeID** in Employees.
4. **Queries-Customers**: One-to-Many from Customers to Queries.
    - A customer can raise many queries, but each query is raised by one customer.
    - Foreign Key: **CustomerID** in Queries references **CustomerID** in Customers.
5. **Queries-Employees**: One-to-Many from Employees to Queries.
    - An employee can address many queries, but each query is addressed by one employee.
    - Foreign Key: **EmployeeID** in Queries references **EmployeeID** in Employees.
6. **Events-Customers**: Many-to-Many from Events to Customers.
    - A customer can attend many events, and each event is visited by many customers.


Title
ItemID
Author/Artist
Publication year
Genre
Items
Stock
Availability
Stock
ItemID
Loan Date
Due Date
Loans
Return Date
CustomerID
Loan ID
F_ItemID
Author/Artist
Publication year
Genre
Future Items
Title
Stock
Availability Date
QueryID
CustomerID
Query Text
Status
Queries
EmployeeID
Name
EmployeeID
Employees
Type
EmployeeID
Event ID
Room Number
Number of Attendees
Events
Max Attendees
Name
Audicence
Price
CustomerID
Name
Customers
Balance
Number of loans

Anomalies

Identifying Functional Dependencies (FDs):

1. **Items**:
    - **ItemID → Type, Title, AuthorArtist, PublicationYear, Genre, Availability, Stock**
        - This suggests that the **ItemID** uniquely determines all other attributes in the table.
2. **Customers**:
    - **CustomerID → Name, NumberOfLoans, Balance**
        - The **CustomerID** uniquely determines the name, number of loans, and balance of the customer.
3. **Loans**:
    - **LoanID → CustomerID, ItemID, LoanDate, DueDate, ReturnDate**
        - Each loan has a unique **LoanID** that determines the details of the loan.
    - Note: There isn't a direct functional dependency between **CustomerID** and **ItemID** because a customer can borrow many items and an item can be borrowed by many customers over time.
4. **Employees**:
    - **EmployeeID → Name, Type**
        - The **EmployeeID** uniquely determines the name and type of the employee.
5. **Events**:
    - **EventID → EmployeeID, RoomNumber, Name, Audience, Price, Number_of_Attendees, MAX_Attendees**
        - The **EventID** determines all details of an event.
6. **FutureItems**:
    - **F_ItemID → Type, Title, AuthorArtist, PublicationYear, Genre, AvailabilityDate, Stock**

- Similar to Items, the **F_ItemID** uniquely identifies upcoming items.

7. **Queries**:
    - **QueryID → CustomerID, QueryText, EmployeeID, Status**
        - Each query can be uniquely identified using the **QueryID**.

Checking BCNF:

For a table to be in BCNF:

1. It must be in 3NF.
2. For every non-trivial functional dependency $X \rightarrow Y$, X should be a superkey.

By examining the FDs for each table, we see that the left side of every FD (like **ItemID**, **CustomerID**, **LoanID**, etc.) is indeed a superkey for their respective tables. Therefore, the design appears to be in BCNF.

Proving No Bad FDs:

To prove that there are no bad functional dependencies, we should ensure:

1. No partial dependencies: No attribute is functionally dependent on a part of a primary key. Given that all our tables with composite keys (like **Loans**) don't have attributes depending solely on a part of the key, we can confirm there are no partial dependencies.
2. No transitive dependencies: Non-key attributes shouldn't determine other non-key attributes. By examining the tables, we don't see such dependencies, which means there are no transitive dependencies.