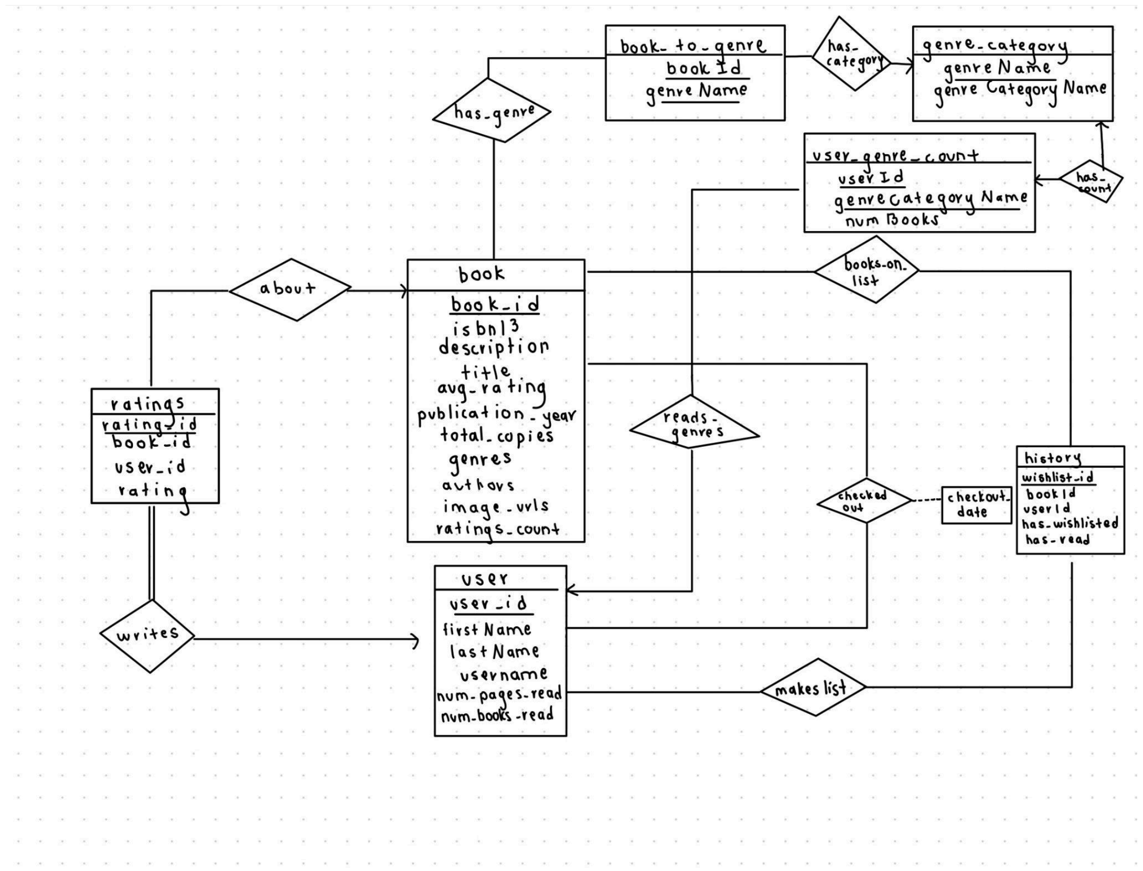
1. **ER Diagram:**



2. **Result Relations After Converting the ER Diagram to Relations:**

user (

userId int,

username varchar(255) not null,

password varchar(255) not null,

firstName varchar(255) not null,

lastName varchar(255) not null,

num_pages_read int,

num_books_read

primary key (userId),

unique (username),

```
);
book (
        bookId int,
        title varchar(255) not null,
        authors varchar(255) not null,
        isbn13 varchar(13) not null unique,
        description text,
        genres varchar(255) not null,
        average_rating float,
        original_publication_year int,
        ratings_count int,
        image_url varchar(255),
        total_copies int not null,
        page_count int
        primary key(bookId)
);
ratings (
        rating_id int,
        bookId int not null,
        userId int not null,
        rating int not null,
        primary key (rating_id),
        unique (bookId, userId),
        foreign key (bookId) references book(bookId),
        foreign key (userId) references user(userId)
);
```

```
curr_checkout (
        userId int,
        bookId int,
        checkout_date date not null,
        primary key (userId, bookId),
        foreign key (userId) references user(userId),
        foreign key (bookId) references book(bookId)
);
history (
        wishlist_id int,
        bookId int,
        userId int,
        has_wishlisted boolean not null,
        has_read boolean not null,
        unique (bookId, userId),
        primary key (wishlist_id),
        foreign key (bookId) references book(bookId),
        foreign key (userId) references user(userId)
);
genre_category (
        genreName varchar(100),
        genreCategoryName varchar(100) not null,
        primary key (genreName)
);
book_to_genre (
        bookId int,
	genreName varchar(100),

	primary key (bookId, genreName),

	foreign key(bookId) references book(bookId),

	foreign key(genreName) references genre_category(genreName)

);

user_genre_count (

	userId int,

	genreCategoryName varchar(100),

	numBooks int,

	primary key (userId, genreCategoryName),

	foreign key (userId) references user(userId)

);
```

3. **Functional Dependencies:**

   fd1: userId → firstName, lastName, username

   fd2: bookId → isbn13, description, genres, title, authors, average_rating, original_published_date, ratings_count, image_url, total_copies, page_count

   fd3: bookId, userId → rating

4. **BCNF Normalization Steps and Final Normalized Relations:**

   BCNF Decomposition:

   Attributes: (isbn13, description, genres, bookId, title, authors, average_rating, original_published_date, ratings_count, image_url, userId, rating, firstName, lastName, username, total_copies, page_count)

   Decomposition Steps:

   Step 1: Check if fd1 is in BCNF  (userId → firstName, lastName, username)

   - Since userId is not a superkey, we decompose:

R1: (userId, firstName, lastName, username)

R2: (userId, bookId, isbn13, description, genres, title, authors, average_rating, original_published_date, ratings_count, image_url, total_copies, rating, page_count)

Step 2: Check if fd2 is in BCNF (bookId → book attributes)

- Since book_id is not a superkey in R2, decompose R2 into:

R3: (bookId, isbn13, description, genres, title, authors, average_rating, original_published_date, ratings_count, image_url, total_copies, page_count)

R4: (bookId, userId, rating)

Step 3: Check if fd3 is in BCNF: (bookId, userId → rating)

- This determinant is a superkey for R4; therefore, R4 is already in BCNF

Final BCNF Relations:

R1: (userId, firstName, lastName, username)

R3: (bookId, isbn13, description, genres, title, authors, average_rating, original_published_date, ratings_count, image_url, total_copies, page_count)

R4: (book_id, user_id, rating)