Team Members:
Shang Ke and Tengri Zhang

Project Description:
Our team has chosen to create a simple library management system. We will create a database of books, movies, albums and patrons, and allow librarians as users to keep track of all the books, movies and albums in stock and those checkout by patrons. Our database will also provide some simple search functions for users. Our team has chosen to use a simple user interface that provides all our currently supported functionality on a single web page. We implement the Web front-end of our application in python using Psycopg2 and Streamlit library. See below for more information about the sets and rules in our database.

Entity sets, Relationship sets, Business Rules:

Book:
- The Book set has primary key: ISBN
- The other attributes are Name, Year, Genre, Publisher and In_Stock
- A Book is written by at least one Author

Author:
- The Author set has primary key: Author_ID
- The other attributes are Name and Country
- An Author writes at least one Book

Patron:
- The Patron set has primary key: Patron_ID
- The other attributes are Name, Address, Email and Phone

Album:
- The Album set has primary key: Album_ID
- The other attributes are Name, Year, Genre and In_Stock
- An Album is created by at least one Artist
- An Album contains at least one Track

Track:
- The Track set is a weak entity set
- The Track set has identifying owner: Album
- The Track set has partial key: Name
- A track belongs to exactly one Album

Artist:
- The Artist set has primary key: Artist_ID

- The other attributes are Name and Country
- An Artist creates at least one Album

Movie:
- The Movie set has primary key: Movie_ID
- The other attributes are Name, Year, Genre, Director and In_Stock
- A Movie has at least one Actor

Actor:
- The Actor set has primary key: Actor_ID
- The other attributes are Name and Country
- An Actor acts in at least one Movie

checkout_by:
- The three checkout_by sets are relationship sets
- They all have attributes Number and Date

Data Acquisition:

The data loaded into the database will be collected from NYU Libraries, IMDb, Billboard, and Wikipedia. The data will be manually generated in csv format and then loaded into the database with the use of the following Postgres “copy” command:

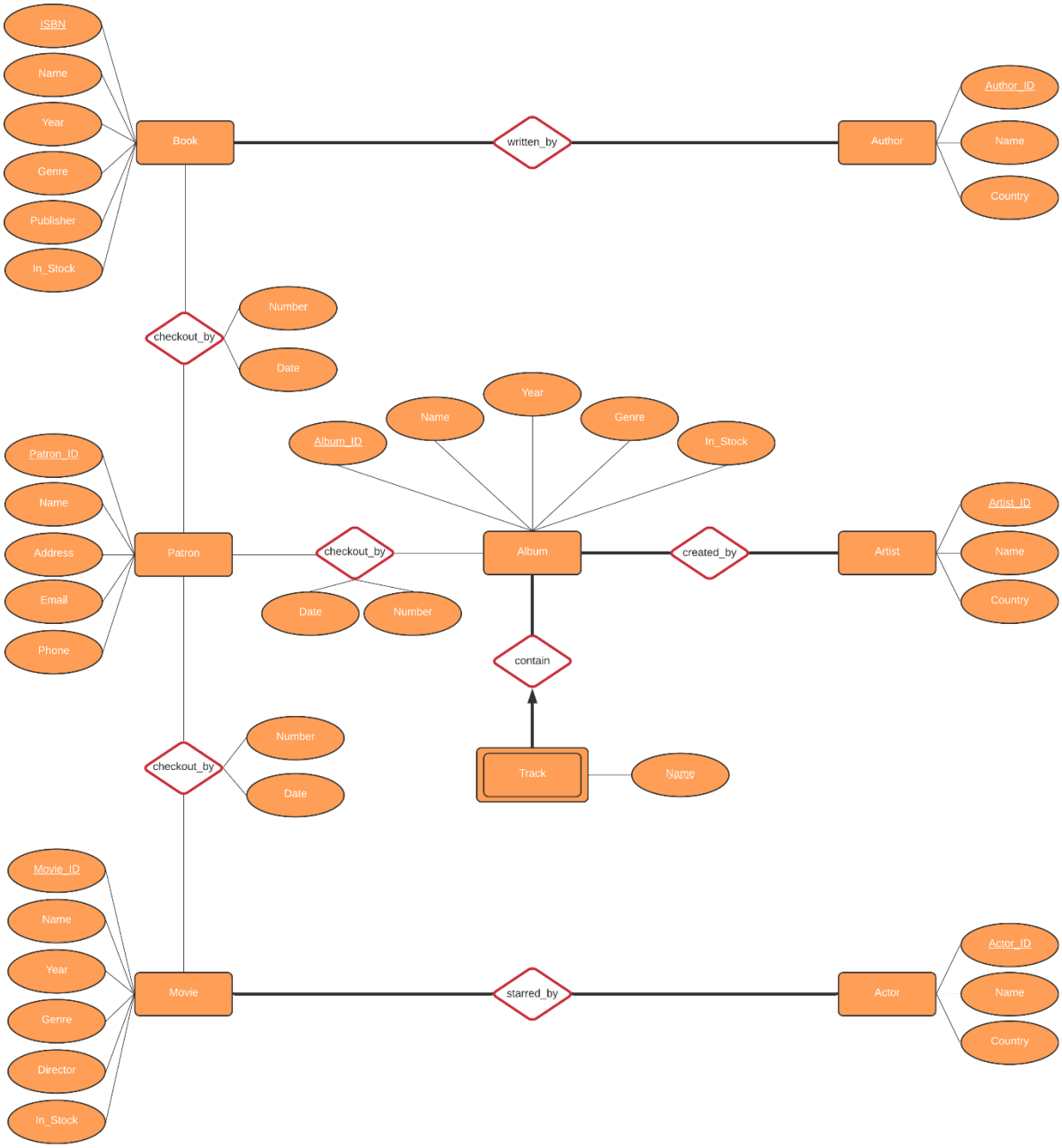
cat authors.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY authors from STDIN CSV HEADER"
cat actors.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY actors from STDIN CSV HEADER"
cat albums.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY albums from STDIN CSV HEADER"
cat artists.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY artists from STDIN CSV HEADER"
cat patrons.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY patrons from STDIN CSV HEADER"
cat books.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY books from STDIN CSV HEADER"
cat movies.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY movies from STDIN CSV HEADER"
cat album_tracks.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY album_tracks from STDIN CSV HEADER"
cat album_check.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY album_check from STDIN CSV HEADER"
cat book_check.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY book_check from STDIN CSV HEADER"
cat movie_check.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY movie_check from STDIN CSV HEADER"

cat starred_by.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY starred_by from STDIN CSV HEADER"

cat written_by.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY written_by from STDIN CSV HEADER"

cat created_by.csv | psql -U tz805 -d tz805-db -h localhost -p 5432 -c "COPY created_by from STDIN CSV HEADER"

ER Diagram:


ISBN
Name
Year
Genre
Publisher
In_Stock
Book
written_by
Author
Author_ID
Name
Country
checkout_by
Number
Date
Year
Name
Genre
Album_ID
In_Stock
Patron_ID
Name
Address
Email
Phone
Patron
checkout_by
Album
created_by
Artist
Artist_ID
Name
Country
Date
Number
contain
checkout_by
Number
Date
Track
Name
Movie_ID
Name
Year
Genre
Director
In_Stock
Movie
starred_by
Actor
Actor_ID
Name
Country

SQL File:

```
drop table if exists books cascade ;
drop table if exists authors cascade ;
drop table if exists written_by cascade;
drop table if exists patrons cascade;
drop table if exists book_check cascade;
drop table if exists movies cascade;
drop table if exists actors cascade;
drop table if exists starred_by cascade;
drop table if exists movie_check cascade;
drop table if exists albums cascade;
drop table if exists tracks cascade;
drop table if exists artists cascade;
drop table if exists created_by cascade;
drop table if exists album_check cascade;

create table books (
   isbn       varchar(13) primary key,
   name       varchar(64),
   year       integer,
   genre      varchar(16),
   publisher   varchar(64),
   stock    integer
);

create table authors (
   id     integer primary key,
   name    varchar(32),
```

```
    country varchar(32)
);

create table written_by (
    isbn    varchar(13),
    author  integer,
    primary key(isbn,author),
    foreign key(isbn) references books(isbn),
    foreign key(author) references authors(id)
);

create table patrons (
    id          integer primary key,
    name        varchar(32),
    address     varchar(64),
    email       varchar(64),
    phone       char(12)
);

create table book_check (
    id      integer,
    isbn    varchar(13),
    date    date,
    number  integer,
    primary key (id,isbn,date),
    foreign key(id) references patrons(id),
    foreign key(isbn) references books(isbn)
);
```

```
create table movies (
   id         integer primary key,
   name       varchar(64),
   year       integer,
   genre      varchar(16),
   director   varchar(32),
   stock   integer
);

create table actors (
   id     integer primary key,
   name   varchar(32),
   country varchar(32)
);

create table starred_by (
   mid    integer,
   aid    integer,
   primary key(mid,aid),
   foreign key(mid) references movies(id),
   foreign key(aid) references actors(id)
);

create table movie_check (
   pid    integer,
   mid    integer,
   date   date,
```

```
    number  integer,
    primary key (pid,mid,date),
    foreign key(pid) references patrons(id),
    foreign key(mid) references movies(id)
);

create table albums (
    id          integer primary key,
    name        varchar(64),
    year        integer,
    genre       varchar(16),
    stock    integer
);

create table album_tracks (
    id      integer,
    name    varchar(64),
    primary key(id,name),
    foreign key (id) references albums(id) on delete cascade
);

create table artists (
    id      integer primary key,
    name    varchar(32),
    country varchar(32)
);

create table created_by (
```

```
    artist    integer,
    album     integer,
    primary key(album,artist),
    foreign key(artist) references artists(id),
    foreign key(album) references albums(id)
);

create table album_check (
    pid     integer,
    aid     integer,
    date    date,
    number  integer,
    primary key (pid,aid,date),
    foreign key(pid) references patrons(id),
    foreign key(aid) references albums(id)
);
```