QwikTix

Fandango meets IMDB meets Apple’s TV app

Sarina Dass
Kirk Morgan
Viviano Cantu
Zongru Wang

Project Motivation

Large database full of information about movies, theaters, and streaming services

Use AI to help users find new movies

Provide an all-in-one place for users to learn about and watch movies

System Description

What’s in our database?

Users (with view history)

Movies (with credits for actors, directors, etc.)

Movie theaters (with orders)

Streaming services (with orders)

What can users do?

Make an account

“Love” a movie

Order a movie ticket from a local theater

See what movies they’ve ordered

Find out what movies they loved but didn’t order

Keeping track of credits

Credit an existing actress for a movie

Find all movies directed by a particular person

Find all people credited for a particular movie

Finding out what’s popular

Top 10 studios (by revenue)

Top 3 genres (by revenue)

Reports

Create a list of names of people and the number of their movies which are loved by users

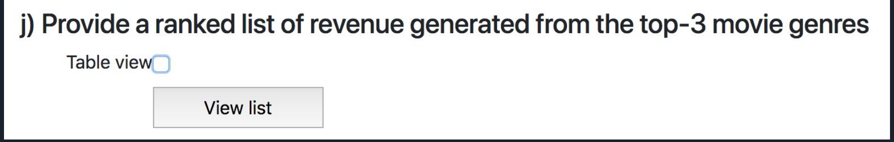
Find number of orders from all theaters and all streaming services

Find all people with a given role who have been in movies with different genres

Find all movies that were loved or ordered by a particular user

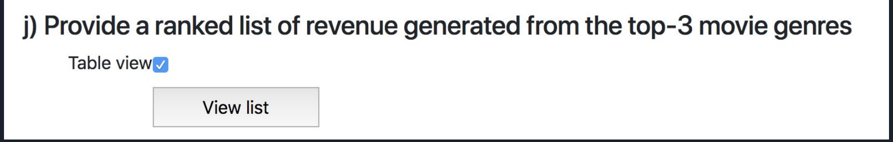
Find the most loved movies in every ZIP code where at least one person lives in a given country

User Interface



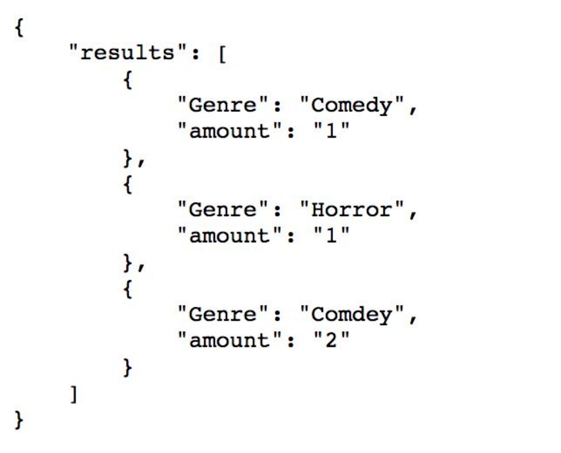
j) Provide a ranked list of revenue generated from the top-3 movie genres

Table view□

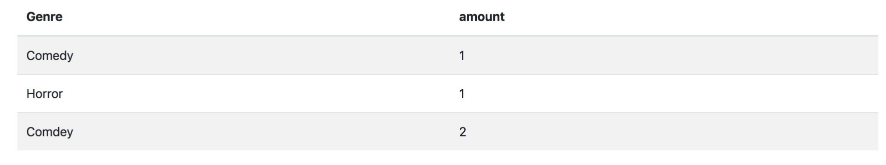


j) Provide a ranked list of revenue generated from the top-3 movie genres

Table view☑



```
{
    "results": [
        {
            "Genre": "Comedy",
            "amount": "1"
        },
        {
            "Genre": "Horror",
            "amount": "1"
        },
        {
            "Genre": "Comdey",
            "amount": "2"
        }
    ]
}
```



Genre	amount
Comedy	1
Horror	1
Comdey	2

System Architecture

What tech did we use?

Back-End:

- MySQL
- Python 2.7
  - PyMySQL
  - Flask
- Bcrypt

Front-End:

- HTML
- CSS

/taskb?UserID=10&MovieID=2&format=table

/taskb?UserID=10&MovieID=2
OR
/taskb?UserID=10&MovieID=2&format=json

Security

Escaping Queries

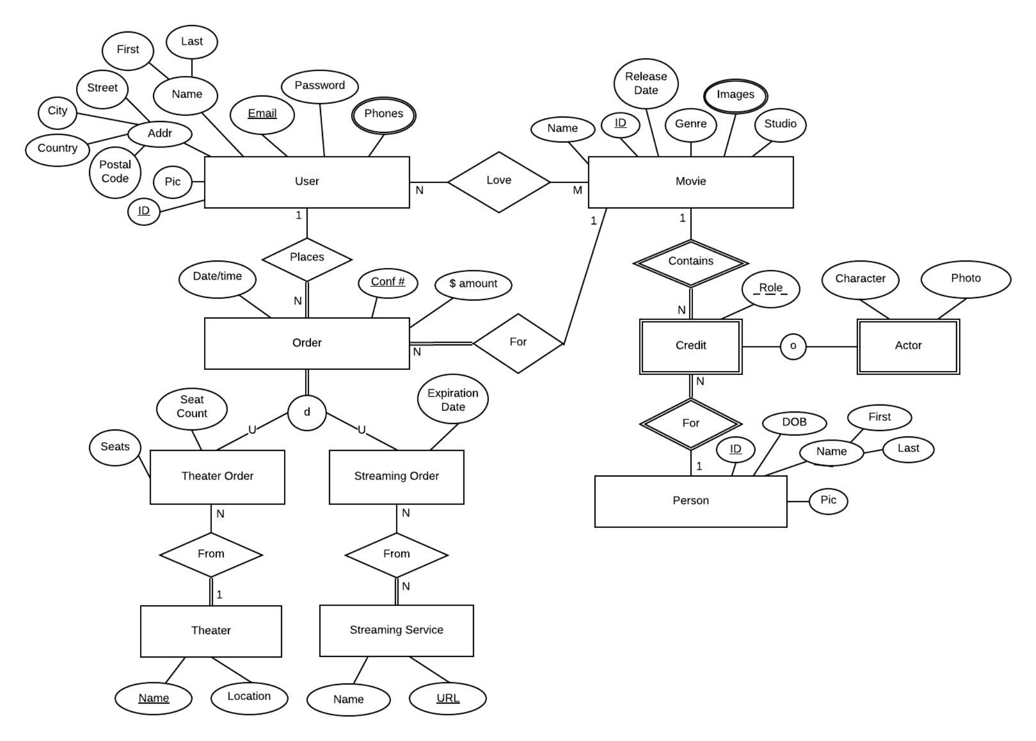
```
a = """INSERT INTO User (FirstName, LastName, Email, Password, Picture, Street, City, PostalCode, Country)
         VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s);"""
```

Encrypting and Salting Passwords

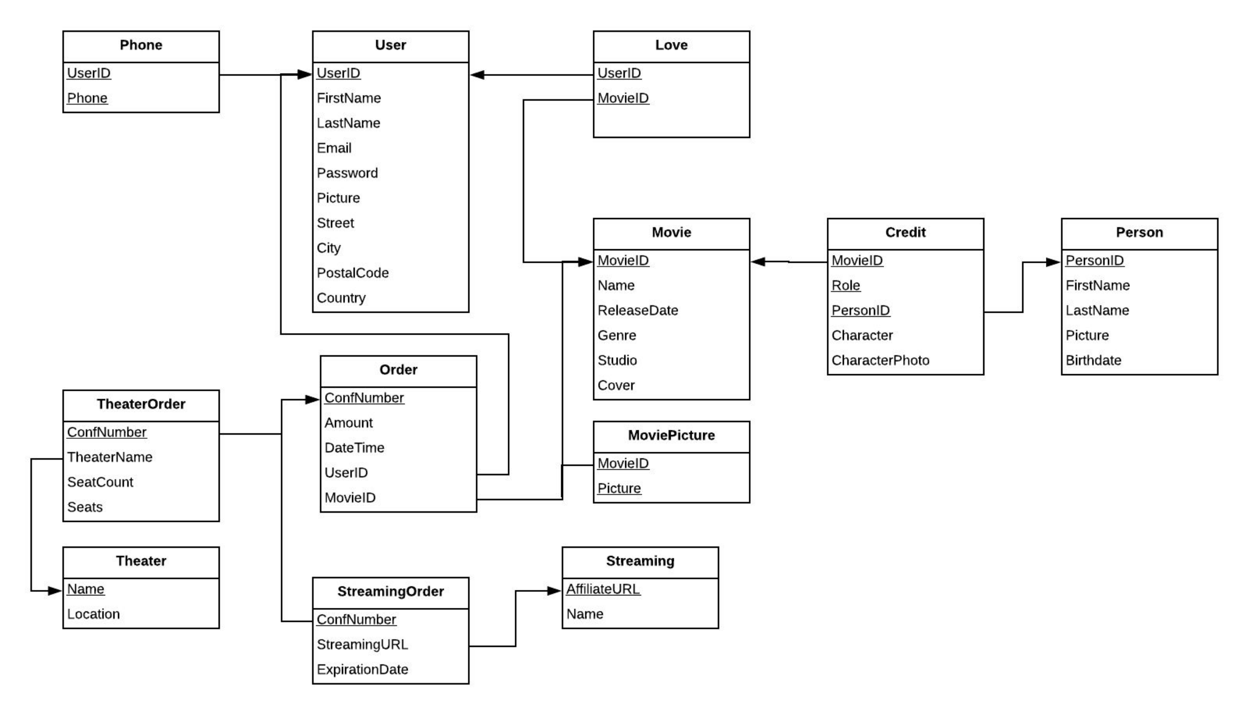
```
bcrypt.hashpw(x.encode('utf8'), bcrypt.gensalt())
```

Database Design

ERD


First
Last
Street
Name
City
Addr
Country
Postal Code
Pic
ID
Email
Password
Phones
User
N
Love
M
Movie
Name
ID
Release Date
Genre
Images
Studio
1
Places
N
Date/time
Conf #
$ amount
Order
N
For
1
1
Contains
N
Role
Character
Photo
Credit
o
Actor
N
For
1
ID
DOB
First
Name
Last
Person
Pic
d
U
U
Seat Count
Seats
Theater Order
Expiration Date
Streaming Order
N
From
1
Theater
Name
Location
N
From
N
Streaming Service
Name
URL

Relational Model



Phone
UserID
Phone

User
UserID
FirstName
LastName
Email
Password
Picture
Street
City
PostalCode
Country

Love
UserID
MovieID

Movie
MovieID
Name
ReleaseDate
Genre
Studio
Cover

Credit
MovieID
Role
PersonID
Character
CharacterPhoto

Person
PersonID
FirstName
LastName
Picture
Birthdate

TheaterOrder
ConfNumber
TheaterName
SeatCount
Seats

Order
ConfNumber
Amount
DateTime
UserID
MovieID

MoviePicture
MovieID
Picture

Theater
Name
Location

StreamingOrder
ConfNumber
StreamingURL
ExpirationDate

Streaming
AffiliateURL
Name

Data Sources

QwikTix is still in testing

Data is currently sample data

Physical Design

Currently in 3NF

Use cases don’t currently require denormalization or indexing beyond primary keys

Having ID’s (for users and movies) as the primary key has been sufficiently fast so far