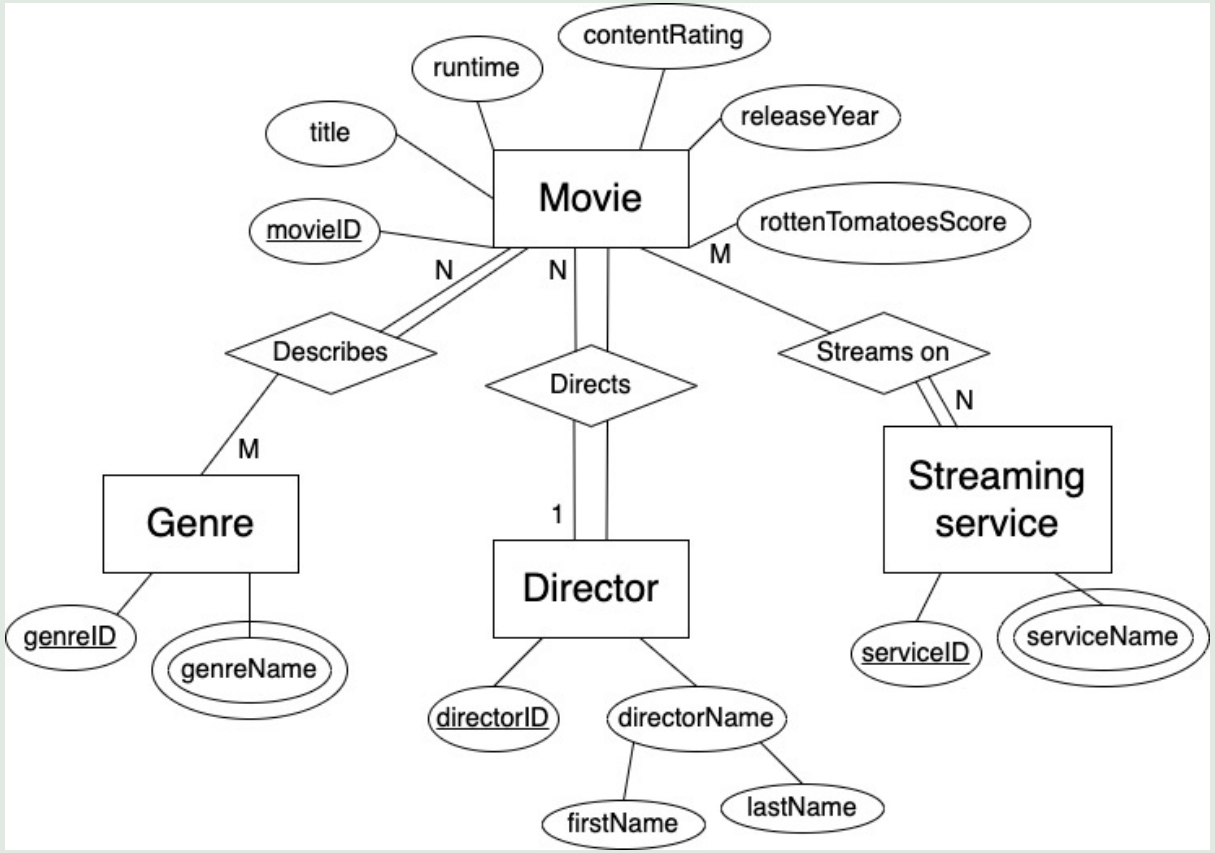
Database on Movies

by Mari Kwee and Quinn Goldstein

Project description

- Database for movies and other information about them
- Faced with the problem of not knowing what to watch on Netflix and spending more time browsing and not enough time watching
- Streaming services do not tell you everything about a movie
- Streamline the process of finding what to watch

Entity-relationship and relational models



Movie (movieID, title, runtime, contentRating, releaseYear, rottenTomatoesScore, directorID) [fk: directorID references Director.directorID]
Genre (genreID, genreName)
Director (directorID, firstName, lastName)
StreamingService (serviceID, serviceName)
Describes (movieID, genreID) [fk: movieID references Movie.movieID, genreID references Genre.genreID]
StreamsOn (movieID, serviceID) [fk: movieID reference Movie.movieID, serviceID references StreamingService.serviceID]

Implementation of relations model using SQL

Step 1: Drop tables if they exist

```
1  use bw_db2;
2
3  # drop tables if they exist
4  drop table if exists streamsOn;
5  drop table if exists describes;
6  drop table if exists movie;
7  drop table if exists streamingService;
8  drop table if exists director;
9  drop table if exists genre;
10
```

Step 2: Create Tables

```
12      # create tables
13  create table genre (
14          genreID         integer,
15          genreName       varchar(40)     not null,
16          primary key (genreID)
17      );
18
```

```
19  create table director (
20          directorID  integer,
21          firstName   varchar(40),
22          lastName    varchar(40),
23          primary key (directorID)
24      );
```

```
create table movie (
    movieID             integer auto_increment,
    title               varchar(80)     not null,
    runtime             integer,
    contentRating       varchar(10),
    releaseYear         integer,
    rottenTomatoesScore integer,
    directorID          integer,
    primary key (movieID),
    foreign key (directorID) references director (directorID)
        on delete CASCADE on update CASCADE
);
```

```
create table describes (
    movieID     integer,
    genreID     integer,
    primary key (movieID, genreID),
    foreign key (movieID) references movie (movieID)
        on delete CASCADE on update CASCADE,
    foreign key (genreID) references genre (genreID)
        on delete CASCADE on update CASCADE
);
```

```
create table streamingService (
    serviceID   integer,
    serviceName varchar(40),
    primary key (serviceID)
);
```

```
create table streamsOn (
    movieID     integer,
    serviceID       integer,
    primary key (movieID, serviceID),
    foreign key (movieID) references movie (movieID)
        on delete CASCADE on update CASCADE,
    foreign key (serviceID) references streamingService (serviceID)
        on delete CASCADE on update CASCADE
);
```

```
5 •  desc genre;
7 •  desc director;
8 •  desc streamingService;
9 •  desc movie;
0 •  desc describes;
1 •  desc streamsOn;
```

Field	Type	Null	Key	Default	Extra
movieID	int	NO	PRI	NULL	auto_increment
title	varchar(80)	NO		NULL	
runtime	int	YES		NULL	
contentRating	varchar(10)	YES		NULL	
releaseYear	int	YES		NULL	
rottenTomatoesScore	int	YES		NULL	
directorID	int	YES	MUL	NULL	

Field	Type	Null	Key	Default	Extra
genreID	int	NO	PRI	NULL	
genreName	varchar(40)	NO		NULL	

Field	Type	Null	Key	Default	Extra
movieID	int	NO	PRI	NULL	
genreID	int	NO	PRI	NULL	

Field	Type	Null	Key	Default	Extra
directorID	int	NO	PRI	NULL	
firstName	varchar(40)	YES		NULL	
lastName	varchar(40)	YES		NULL	

Field	Type	Null	Key	Default	Extra
movieID	int	NO	PRI	NULL	
serviceID	int	NO	PRI	NULL	

Field	Type	Null	Key	Default	Extra
serviceID	int	NO	PRI	NULL	
serviceName	varchar(40)	YES		NULL	

Step 3: Insert tuples into a table

```
insert into director (directorID, firstName, lastName)
    values (1, "Martin", "Scorsese"),
    (2, "Hayao", "Miyazaki"),
    (3, "Stanley", "Kubrick"),
    (4, "Chloé", "Zhao"),
    (5, "Jordan", "Peele");
```

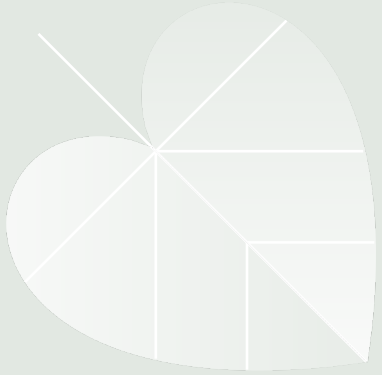
Step 4: Retrieve the data from a table

```
select *
from director
```

directorID	firstName	lastName
1	Martin	Scorsese
2	Hayao	Miyazaki
3	Stanley	Kubrick
4	Chloé	Zhao
5	Jordan	Peele
NULL	NULL	NULL

Populating data

- Manually gathered data about movies using Google and compiled into Excel
- Built other tables using directors, genres, and streaming services of these selected ~300 movies
- Auto-generated hundreds of other director names to demonstrate scope and functionality of database using Mockaroo
- Loaded in data to MySQL with .csv files



Tables: movie and director

movieID	title	runtime	contentRating	releaseYear	rottenTomatoesScore	directorID
001	Star Wars: The Phantom Menace	136	PG	1999	52	58
002	Star Wars: Attack of the Clones	120	PG	2002	65	58
003	Star Wars: Revenge of the Sith	140	PG-13	2005	80	58
004	Star Wars: A New Hope	105	PG	1977	92	58
005	Star Wars: The Empire Strikes Back	124	PG	1980	94	49
006	Star Wars: Return of the Jedi	131	PG	1983	82	60
007	Star Wars: The Force Awakens	136	PG-13	2015	93	01
008	Star Wars: The Last Jedi	152	PG-13	2017	91	46
009	Star Wars: The Rise of Skywalker	142	PG-13	2019	52	01

directorID	firstName	lastName
01	JJ	Abrams
02	Paul Thomas	Anderson
03	Wes	Anderson
04	Darren	Aronofsky
05	David	Ayer
06	Michael	Bay
07	Ingmar	Bergman
08	Kathryn	Bigelow
09	Shane	Black
10	Anna	Boden
11	Joon-ho	Bong
12	Danny	Boyle
13	Kenneth	Branagh
14	Bo	Burnham
15	Tim	Burton
16	James	Cameron
17	Damien	Chazelle
18	Jon M.	Chu
19	Ethan, Joel	Coen Brothers
20	Chris	Columbus

Tables: movie, streamingService, streamsOn

movieID	title	runtime	contentRating	releaseYear	rottenTomatoesScore	directorID
001	Star Wars: The Phantom Menace	136	PG	1999	52	58
002	Star Wars: Attack of the Clones	120	PG	2002	65	58
003	Star Wars: Revenge of the Sith	140	PG-13	2005	80	58
004	Star Wars: A New Hope	105	PG	1977	92	58
005	Star Wars: The Empire Strikes Back	124	PG	1980	94	49
006	Star Wars: Return of the Jedi	131	PG	1983	82	60
007	Star Wars: The Force Awakens	136	PG-13	2015	93	01
008	Star Wars: The Last Jedi	152	PG-13	2017	91	46
009	Star Wars: The Rise of Skywalker	142	PG-13	2019	52	01

serviceID	serviceName
01	Netflix
02	HBO Max
03	Disney+
04	Apple TV+
05	Peacock
06	Hulu
07	Amazon Prime Video
08	Paramount+
09	Rental

movieID	serviceID
001	03
002	03
003	03
004	03
005	03
006	03
007	03
008	03
009	03

Tables: movie, genre, describes

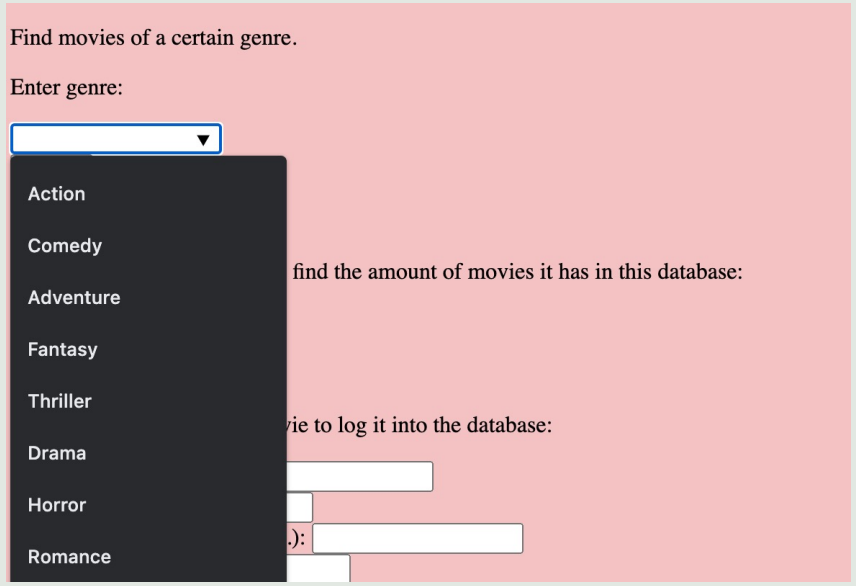
movieID	title	runtime	contentRating	releaseYear	rottenTomatoesScore	directorID
001	Star Wars: The Phantom Menace	136	PG	1999	52	58
002	Star Wars: Attack of the Clones	120	PG	2002	65	58
003	Star Wars: Revenge of the Sith	140	PG-13	2005	80	58
004	Star Wars: A New Hope	105	PG	1977	92	58
005	Star Wars: The Empire Strikes Back	124	PG	1980	94	49
006	Star Wars: Return of the Jedi	131	PG	1983	82	60
007	Star Wars: The Force Awakens	136	PG-13	2015	93	01
008	Star Wars: The Last Jedi	152	PG-13	2017	91	46
009	Star Wars: The Rise of Skywalker	142	PG-13	2019	52	01

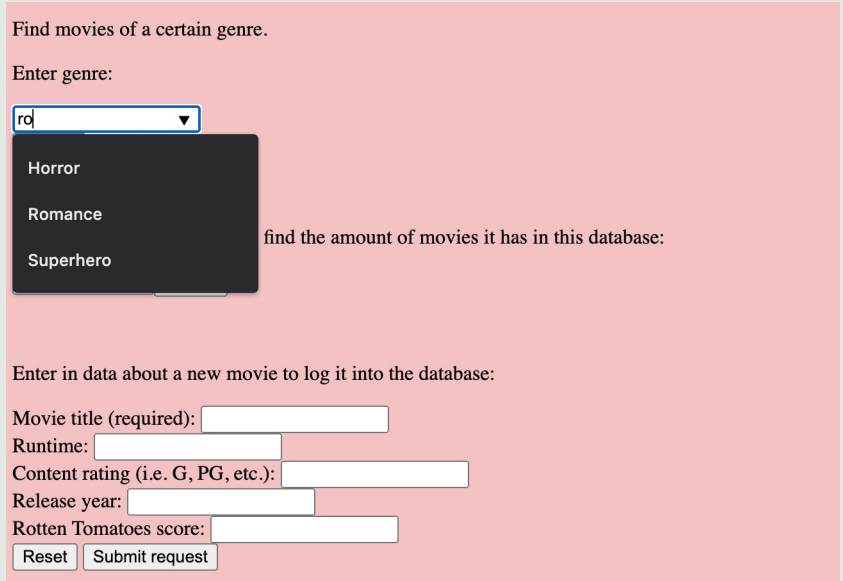
genreID	genreName
01	Action
02	Comedy
03	Adventure
04	Fantasy
05	Thriller
06	Drama
07	Horror
08	Romance
09	Animated
10	Musical
11	Mystery
12	Crime
13	Science fiction
14	Western
15	Documentary

movieID	genreID
001	13
001	03
002	13
002	03
003	13
003	03
004	13
004	03
005	13
005	03
006	13
006	03
007	13
007	03
008	13
008	03
009	13

Application development

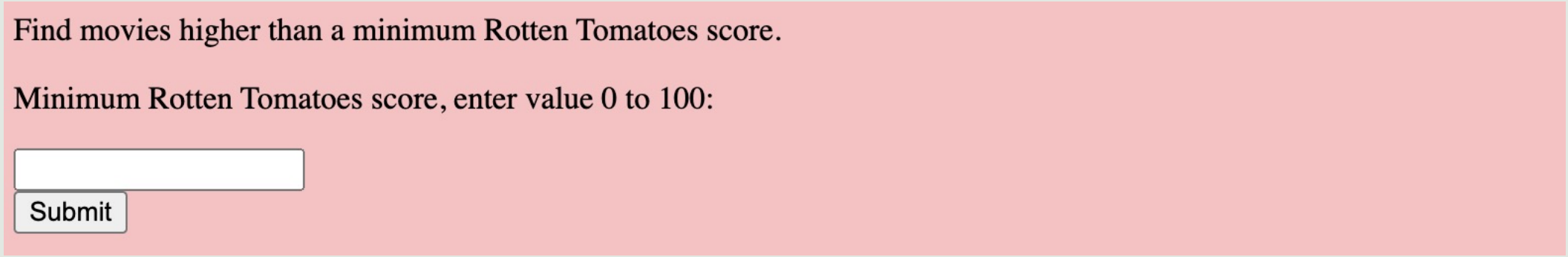
- Developed logical queries and tried to make them user-friendly as possible
- One example included a suggestion list for the query to search movies by genre
  - easier to implement than having multiple names for a similar genre (e.g., entering ‘Science fiction’ vs. ‘Sci-fi’)
  - implemented using HTML data list element





Application development II

- Goal of making queries useful and for further development in the future rather than just fulfilling requirements of a project
- Wanted functionality to reflect an application that could be used by the average person



Challenges throughout the project

- Populating data
  - lengthy, tedious process to manually search up data rather than using some sort of web scraping process
- The web app and using PHP
  - neither of us had used PHP before so there was a learning process
  - implementing some functionality of the web application required change to the database

Achievements and the future of the project

- Find a way to automatically add new released movies into database
- Implement more join queries
  - allow users to find movies of a specific genre on a specific streaming service
  - allow users to specify multiple genres when searching for a movie
  - filter movies by runtime
  - spell check to get valid data
  - allow movie insert to have a director at entry

Thank You