Keisha Arnold
CS 340
Final Project- Movie Database

Outline/Introduction

This is a movie database which show the relationships between a movie, genres, actors, and directors. Being a movie aficionado, this database would be useful as a way to store a large collection of movies- making it easy to browse, search, and organize them by a particular attribute.

A movie will have a relation with three other entities: genres, directors and actors.

A movie can have one to many genres. This is a many-to-many relationship between movies and genres (belong relationship). For example, the movie “Harry Potter and the Sorcerer’s Stone” may have the genres “Family” and “Adventure”. Likewise, a genre can belong to more than one movie, so the movies “Harry Potter” and “Mouse Hunt” can both have the genre “Family”. You also have the ability to remove a genre from a movie (removing from the belong relationship).

A movie can have one to many actors. For example, the movie “Harry Potter and the Sorcerer’s Stone” can have the actors “Emma Watson” and “Daniel Radcliffe”. The relationship between an actor and movie (is_in relationship) will also contain the role(s) or character(s) the actor has played in the movie (char_name). Using the previous example, “Daniel Radcliffe” would have the role of “Harry Potter” and “Emma Watson” would have the role of “Hermione Granger”. It is also possible for an actor to have more than one role in a movie, such is the case with the actor “Benedict Cumberbatch” who plays the roles “Doctor Strange” and “Dormammu” in the movie “Doctor Strange”. You could also have the case where a role/character is in multiple movies, as is the case with sequel-type movies such as “Harry Potter” where some of the characters would be in “Harry Potter and the Sorcerer’s Stone”, “Harry Potter and the Chamber of Secrets”, “Harry Potter and the Prisoner of Azkaban”, etc. The many-to-many relationship between actors and movies has its primary key as the movie_id, actor_id and char_name to account for these situations.

A movie must have only one director, but a director can direct many movies. For example, the director Chris Columbus has directed many movies like “Harry Potter and the Sorcerer’s Stone” and “Home Alone”. In this database, we assume that directors do not also act in movies.

Database Outline

Entities: Properties, …
Movies: Movie_ID, Title, Release_Year, Description, Rating, fk_Dir_ID
Genres: Genre_Type
Belong (M:N relationship between Movies and Genres): b_mid (movie_id), gid (genre_type)
Directors: Director_ID, Dir_fname, Dir_lname, Dir_Gender, Dir_DOB
Actors: Actor_ID, Actor_fname, Actor_lname, Actor_Gender, Actor_DOB
Is_In (M:N relationship between Movies and Actors): mid (movie_id), aid (actor_id), char_name (role/character the actor played in a movie)

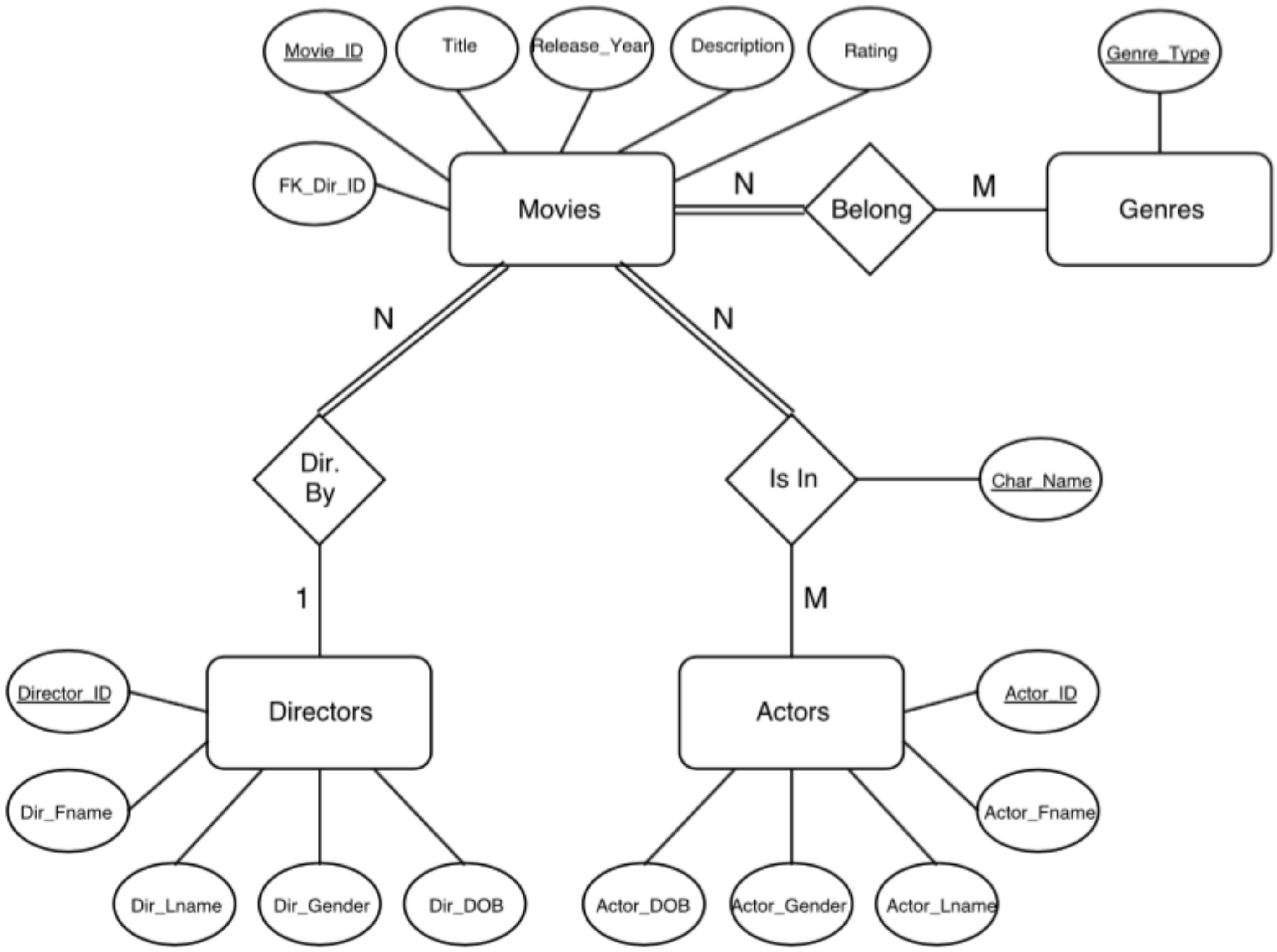
You have a movie database:
A movie is identified by a movie_id (primary key) and has a title, release_year, description, rating, and fk_dir_id.
The movie rating is a number from 1 (terrible) to 5 (excellent).
The fk_dir_id references the dir_id in the directors table.
The combination of title and release_year must be unique.

Each movie has one or more genre types: romantic, comedy, drama, etc. (M:N relationship between movies and genres).
A genre can belong to zero or more movies (so you can have a genre that doesn't belong to any movies).
Each genre_type in the genre entity must be unique and the primary key is the combination of movie_id and genre_type.

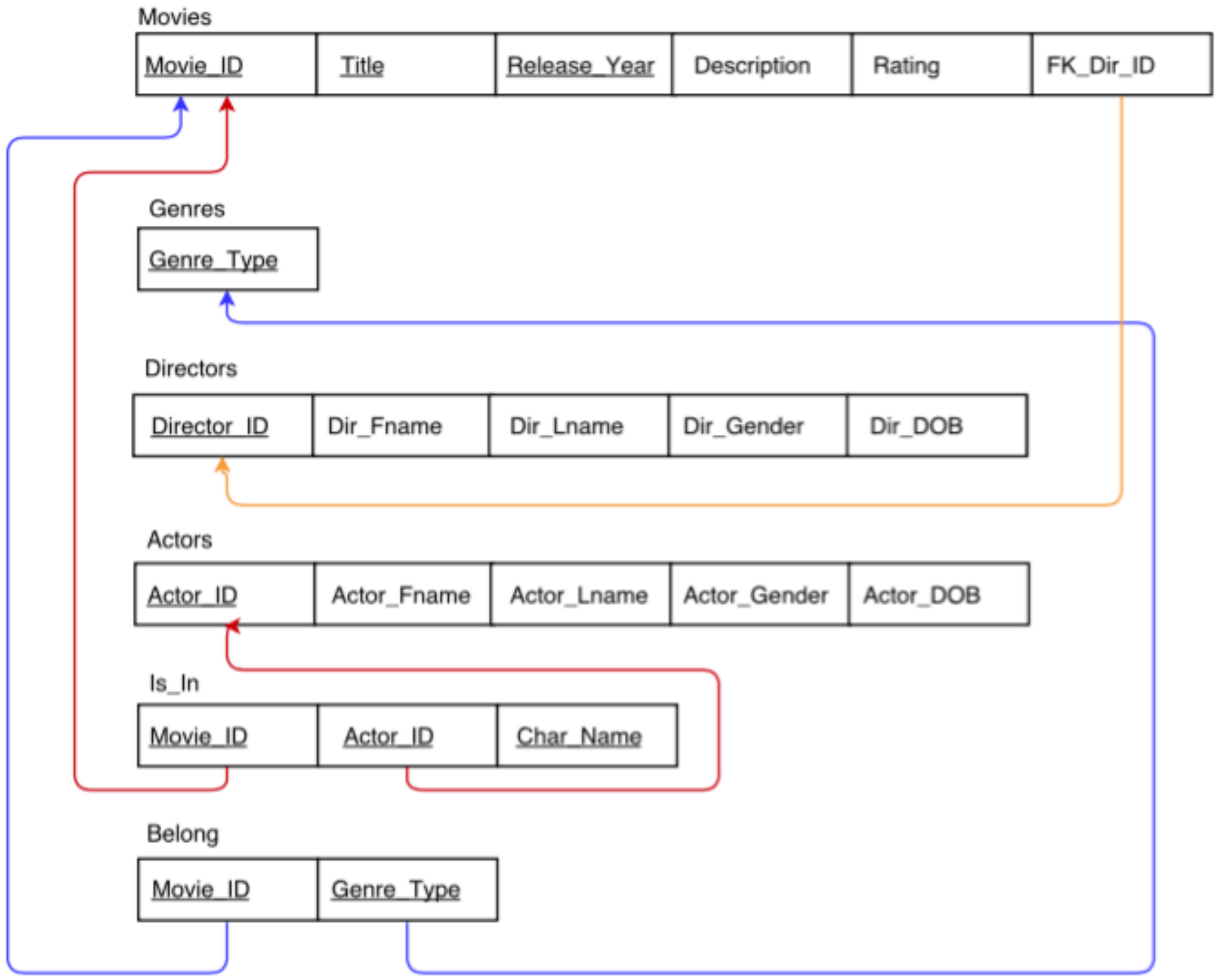
Each movie is directed by exactly one director (1:M relationship between movies and directors).
A director is identified by a dir_id (primary key) and has a dir_fname, dir_lname, dir_gender, and dir_dob.
A director can direct zero or more movies (so you can have a director that doesn't direct any movies).
In this database, we assume that directors do not also act in movies.

Each movie has one or more actors (M:N relationship between movies and actors).
An actor is identified by an actor_id (primary key), and has an actor_fname, actor_lname, gender, and dob.
An actor is in zero or more movies (so you can have an actor that doesn't act in any movies).
Each actor has a character/role they played in a movie (null if they haven't played a character/role). Also, it's possible for one actor to play multiple characters in a single movie. It's also possible to have the same character in multiple movies.
The combination of movie_id, actor_id, and char name must be unique and is the primary key in this many-to-many relationship.

ER Diagram


Movie_ID
Title
Release_Year
Description
Rating
Genre_Type
FK_Dir_ID
Movies
N
Belong
M
Genres
N
N
Dir.
By
Is In
Char_Name
1
M
Director_ID
Directors
Actors
Actor_ID
Dir_Fname
Actor_Fname
Dir_Lname
Dir_Gender
Dir_DOB
Actor_DOB
Actor_Gender
Actor_Lname

Schema


Movies
Movie_ID
Title
Release_Year
Description
Rating
FK_Dir_ID
Genres
Genre_Type
Directors
Director_ID
Dir_Fname
Dir_Lname
Dir_Gender
Dir_DOB
Actors
Actor_ID
Actor_Fname
Actor_Lname
Actor_Gender
Actor_DOB
Is_In
Movie_ID
Actor_ID
Char_Name
Belong
Movie_ID
Genre_Type

Data Definition Queries

```
DROP TABLE IF EXISTS `is_in`;
DROP TABLE IF EXISTS `belong`;
DROP TABLE IF EXISTS `movies`;
DROP TABLE IF EXISTS `genres`;
DROP TABLE IF EXISTS `actors`;
DROP TABLE IF EXISTS `directors`;


CREATE TABLE directors (
        dir_id INT(11) NOT NULL AUTO_INCREMENT,
        dir_fname VARCHAR(225) NOT NULL,
        dir_lname VARCHAR(225) NOT NULL,
        dir_gender VARCHAR(225) NOT NULL,
        dir_dob DATE,
        PRIMARY KEY(dir_id)
) ENGINE=InnoDB;


CREATE TABLE actors (
        actor_id INT(11) NOT NULL AUTO_INCREMENT,
        actor_fname VARCHAR(225) NOT NULL,
        actor_lname VARCHAR(225) NOT NULL,
        actor_gender VARCHAR(225) NOT NULL,
        actor_dob DATE,
        PRIMARY KEY(actor_id)
) ENGINE=InnoDB;


CREATE TABLE genres (
        genre_type VARCHAR(225) NOT NULL,
        PRIMARY KEY(genre_type),
        UNIQUE KEY(genre_type)
) ENGINE=InnoDB;


CREATE TABLE is_in (
        mid INT(11) NOT NULL,
        aid INT(11) NOT NULL,
        char_name VARCHAR(225) NOT NULL,
        PRIMARY KEY(mid, aid, char_name),
        FOREIGN KEY(mid) REFERENCES movies(movie_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
        FOREIGN KEY(aid) REFERENCES actors(actor_id)
        ON UPDATE CASCADE
        ON DELETE NO ACTION
) ENGINE=InnoDB;
```

```
CREATE TABLE belong (
        b_mid INT(11) NOT NULL,
        gid VARCHAR(225) NOT NULL,
        PRIMARY KEY(b_mid, gid),
        FOREIGN KEY(b_mid) REFERENCES movies(movie_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
        FOREIGN KEY(gid) REFERENCES genres(genre_type)
        ON UPDATE CASCADE
        ON DELETE NO ACTION
) ENGINE=InnoDB;

-- populate the tables
-- insert the following into the directors table:

INSERT INTO directors(dir_fname, dir_lname, dir_gender, dir_dob) VALUES
        ("Chris", "Columbus", "Male", “1958-09-10");

INSERT INTO directors(dir_fname, dir_lname, dir_gender, dir_dob) VALUES
        ("Spike", "Jonze", "Male", “1969-10-22”);

INSERT INTO directors(dir_fname, dir_lname, dir_gender, dir_dob) VALUES
        ("Gore", "Verbinski", "Male", “1964-03-16");

INSERT INTO directors(dir_fname, dir_lname, dir_gender, dir_dob) VALUES
        ("Sofia", "Coppola", "Female", “1971-05-14");

INSERT INTO directors(dir_fname, dir_lname, dir_gender, dir_dob) VALUES
        ("Martin", "Campbell", "Male", “1943-10-24");

-- insert the following into the movies table:

INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("Harry Potter and the Sorcerer's Stone", 2001,
        "Adaptation of the first of J.K. Rowlings popular children's novels about Harry Potter",
        4,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Chris" AND d.dir_lname =
        “Columbus"));

INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("Harry Potter and the Chamber of Secrets", 2002,
        "Adaptation of the second of J.K. Rowlings popular children's novels about Harry Potter",
        4,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Chris" AND d.dir_lname =
        “Columbus"));
```

```
INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("Her", 2013,
        "A man becomes fascinated with a new operating system which develops into an
        intuitive and unique entity",
        5,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Spike" AND d.dir_lname =
        “Jonze"));

INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("Mouse Hunt", 1997,
        "Two hapless brothers go to wild extremes to rid their house of a very shrewd mouse",
        3,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Gore" AND d.dir_lname =
        “Verbinski”));

INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("Lost in Translation", 2003,
        "An aging actor Bob Harris, befriends college graduate Charlotte in a Toyko hotel",
        5,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Sofia" AND d.dir_lname =
        "Coppola"));

INSERT INTO movies(title, release_year, description, rating, fk_dir_id) VALUES
        ("The Bling Ring", 2003,
        "Based on a real-world crime ring that preys on wealthy victims",
        3,
        (SELECT d.dir_id FROM directors d WHERE d.dir_fname = "Sofia" AND d.dir_lname =
        “Coppola"));

-- insert the following into the actors table:

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Nathan", "Lane", "Male", “1956-02-03");

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Scarlett", "Johansson", "Female", “1984-11-22");

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Emma", "Watson", "Female", “1990-04-15");

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Daniel", "Radcliffe", "Male", “1989-07-23");

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Bill", "Murray", "Male", "1950-09-21");

INSERT INTO actors(actor_fname, actor_lname, actor_gender, actor_dob) VALUES
        ("Matthew", "Broderick", "Male", “1962-03-21");
```

```
-- insert the following into the genres table

INSERT INTO genres(genre_type) VALUES
	("Drama");

INSERT INTO genres(genre_type) VALUES
	("Romance");

INSERT INTO genres(genre_type) VALUES
	("Adventure");

INSERT INTO genres(genre_type) VALUES
	("Fantasy");

INSERT INTO genres(genre_type) VALUES
	("Family");

INSERT INTO genres(genre_type) VALUES
	("Comedy");

INSERT INTO genres(genre_type) VALUES
	(“Animated");

-- insert the following is_in instances into the is_in table:

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the
	Sorcerer's Stone"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Emma" AND
	a.actor_lname = "Watson"),
	"Hermione Granger");

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the
	Sorcerer's Stone"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Daniel" AND
	a.actor_lname = "Radcliffe"),
	"Harry Potter”);

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the Chamber
	of Secrets"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Emma" AND
	a.actor_lname = "Watson"),
	"Hermione Granger");
```

```
INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the Chamber
	of Secrets"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Daniel" AND
	a.actor_lname = "Radcliffe"),
	"Harry Potter”);

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Her"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Scarlett" AND
	a.actor_lname = "Johansson"),
	“Samantha");

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Mouse Hunt"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Nathan" AND
	a.actor_lname = "Lane"),
	"Ernie Smuntz");

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Lost in Translation"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Scarlett" AND
	a.actor_lname = "Johansson"),
	“Charlotte");

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Lost in Translation"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Bill" AND
	a.actor_lname = "Murray"),
	"Bob Harris");

INSERT INTO is_in(mid, aid, char_name) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "The Bling Ring"),
	(SELECT a.actor_id FROM actors a WHERE a.actor_fname = "Emma" AND
	a.actor_lname = "Watson"),
	“Nicki”);

-- insert the following is_in instances into the belong table:

INSERT INTO belong(b_mid, gid) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the
	Sorcerer's Stone"),
	(SELECT g.genre_type FROM genres g WHERE g.genre_type = "Fantasy"));

INSERT INTO belong(b_mid, gid) VALUES (
	(SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the
	Sorcerer's Stone"),
	(SELECT g.genre_type FROM genres g WHERE g.genre_type = "Adventure"));
```

```
INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the Chamber
        of Secrets"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Fantasy"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Harry Potter and the Chamber
        of Secrets"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Adventure"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Mouse Hunt"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Comedy"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Mouse Hunt"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Family"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Her"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Drama"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Her"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Romance"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Lost in Translation"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Romance"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Lost in Translation"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Drama"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "Lost in Translation"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Comedy"));

INSERT INTO belong(b_mid, gid) VALUES (
        (SELECT m.movie_id FROM movies m WHERE m.title = "The Bling Ring"),
        (SELECT g.genre_type FROM genres g WHERE g.genre_type = "Drama"));
```

Data Manipulation Queries

-- Queries to populate tables:
— movies and directors

```
SELECT m.movie_id, m.title, m.release_year, m.description, m.rating,
    d.dir_fname, d.dir_lname FROM movies m
INNER JOIN directors d ON m.fk_dir_id = d.dir_id
ORDER BY m.title ASC
```

-- actors

```
SELECT * FROM actors;
```

-- directors

```
SELECT * FROM directors;
```

-- genres

```
SELECT * FROM genres;
```

-- Queries with user input:
-- View movies and genres

-- select movie:

```
SELECT m.movie_id, m.title, m.release_year, m.description,
   m.rating, d.dir_fname, d.dir_lname FROM movies m
INNER JOIN directors d ON m.fk_dir_id = d.dir_id
WHERE m.movie_id = [ movieID ];
```

-- select genres belonging to that movie id:

```
SELECT g.genre_type FROM genres g
INNER JOIN belong b ON g.genre_type = b.gid
INNER JOIN movies m ON b.b_mid = m.movie_id
WHERE m.movie_id = [ movieID ];
```

-- View an actors' characters/roles

-- select actor:

```
SELECT a.actor_id, a.actor_fname, a.actor_lname, a.actor_gender,
   a.actor_dob FROM actors a
WHERE a.actor_id = [ actorID ];
```

-- select movie titles and character names for that actor:

```
SELECT m.title, c.char_name FROM is_in c
INNER JOIN actors a ON c.aid = a.actor_id
INNER JOIN movies m ON c.mid = m.movie_id
WHERE a.actor_id = [ actorID ];
```

```
-- filter movies by genre
        -- select genres:
        SELECT * FROM genres;
        -- select movies in that genre:
        SELECT m.movie_id, m.title, m.release_year, m.description,
           m.rating,d.dir_fname, d.dir_lname FROM belong b
        INNER JOIN movies m ON m.movie_id = b.b_mid
        INNER JOIN directors d ON m.fk_dir_id = d.dir_id
        WHERE b.gid = " [ genreFilterID ] ";

-- INSERT queries
-- Add movie (this also adds a director to a movie)
INSERT INTO movies (title, release_year, description, rating,
   fk_dir_id)
VALUES ([ title ], [ release_year ], [ description ], [ rating ],
   [ director ]);

-- Add genre(s) to movie (add to belong relationship)
INSERT INTO belong (b_mid, gid) VALUES ([ movie_id ], [ genre_type ]);

-- Add genre to database
INSERT INTO genres (genre_type) VALUES ([ genre_type ]);

-- Add actor and their role to a movie (add to is_in relationship)
INSERT INTO is_in (mid, aid, char_name) VALUES ([ movie_id ],
   [ actor_id ], [ character_name ]);

-- Add actor to database
INSERT INTO actors (actor_fname, actor_lname, actor_gender, actor_dob)
VALUES ([ actor_fname ], [ actor_lname ], [ actor_gender ],
    [ actor_dob ]);

-- Add director to database
INSERT INTO directors (dir_fname, dir_lname, dir_gender, dir_dob)
VALUES ([ dir_fname ], [ dir_lname ], [ dir_gender ], [ dir_dob ]);

-- DELETE/UPDATE queries
-- Delete movie from database
DELETE FROM movies WHERE movie_id = [ movie_id ];

-- Delete genre(s) from movie (remove from belong relationship)
DELETE FROM belong WHERE b_mid = [ movie_id ] AND gid = [ genre_type ]
```

-- Update/Edit a movie

-- select movie to edit:

```
SELECT * FROM movies WHERE movie_id = [ movie_id ];
```

-- update movie with changes:

-- Note: If the user left a field on the form blank/did not make changes to that field, then the current value for that attribute is kept. If the user entered changes in the form, then the changes will replace the current value for that attribute.

```
UPDATE movies SET title = [ cur_title || edit_tile ], release_year
= [ cur_release_year || edit_release_year ], description =
[ cur_description || edit_description ], rating = [ cur_rating ||
edit_rating ], fk_dir_id = [ cur_director || edit_director ] WHERE
movie_id = [ movie_id ];
```