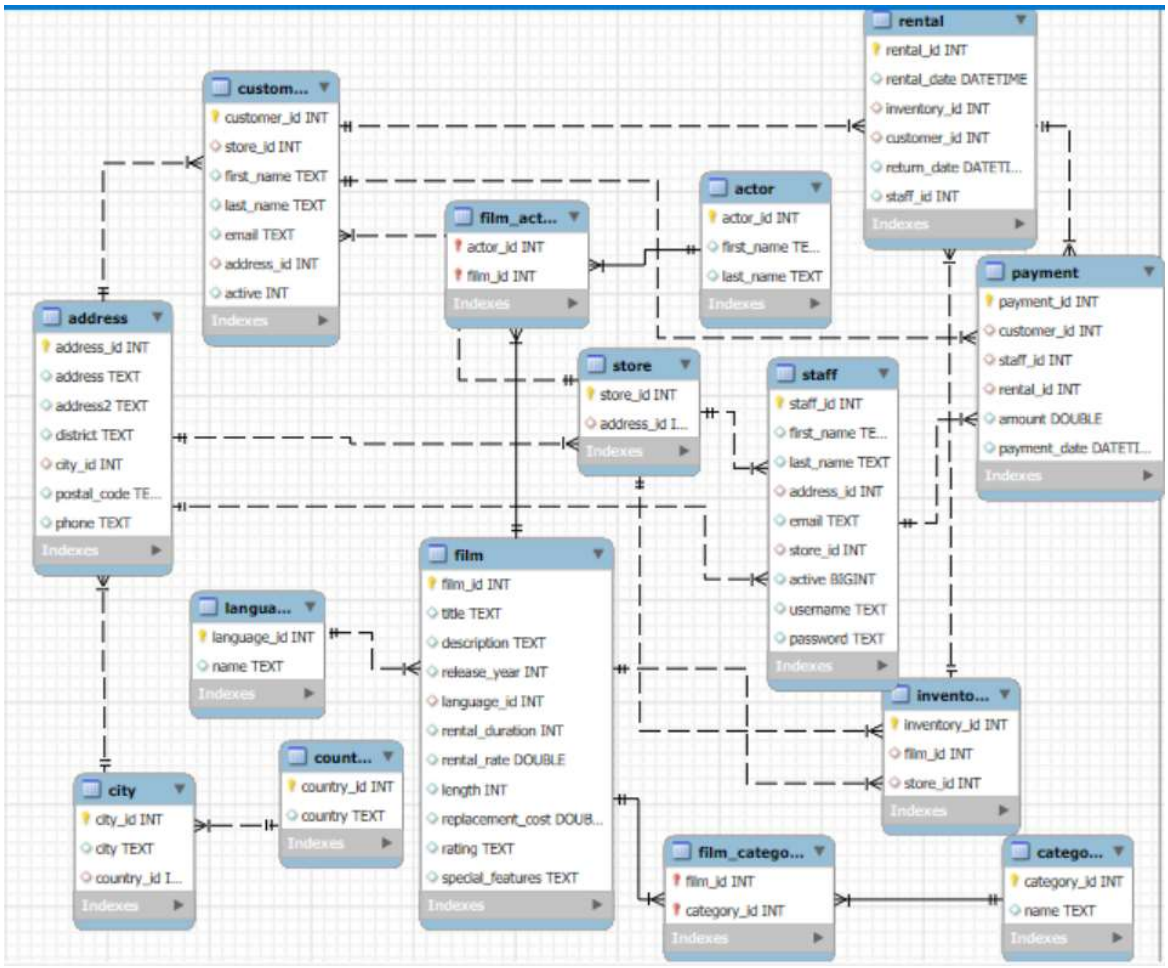
DB Assignment 4

Nick Kraus

11/4/25

- **ER Diagram**



- **Primary and Foreign Keys**

```
-- Setting Primary Keys for Imported Tables
alter table actor add primary key (actor_id);
alter table address add primary key (address_id);
alter table category add primary key (category_id);
alter table city add primary key (city_id);
alter table country add primary key (country_id);
alter table customer add primary key (customer_id);
alter table film add primary key (film_id);
alter table film_actor add primary key (film_id, actor_id);
alter table rental add primary key (rental_id);
alter table staff add primary key (staff_id);
alter table store add primary key (store_id);
alter table film_category add primary key (film_id, category_id);
alter table inventory add primary key (inventory_id);
alter table language add primary key (language_id);
alter table payment add primary key (payment_id);
```

```
-- Setting Foreign Keys for Imported Tables
alter table address add foreign key (city_id) references city (city_id);
alter table city add foreign key (country_id) references country (country_id);
alter table customer add foreign key (store_id) references store (store_id),
  add foreign key (address_id) references address (address_id);
alter table film add foreign key (language_id) references language (language_id);
alter table film_actor add foreign key (film_id) references film (film_id),
  add foreign key (actor_id) references actor (actor_id);
alter table rental add foreign key (customer_id) references customer (customer_id),
  add foreign key (inventory_id) references inventory (inventory_id);
alter table staff add foreign key (address_id) references address (address_id),
  add foreign key (store_id) references store (store_id);
alter table store add foreign key (address_id) references address (address_id);
alter table film_category add foreign key (film_id) references film (film_id),
  add foreign key (category_id) references category (category_id);
alter table inventory add foreign key (film_id) references film (film_id),
  add foreign key (store_id) references store (store_id);
alter table payment add foreign key (customer_id) references customer (customer_id),
  add foreign key (staff_id) references staff (staff_id),
  add foreign key (rental_id) references rental (rental_id);
```

- **Constraints**

```
-- Valid Category Name Constaint
alter table category
   add constraint name_check check (category.name in ("Animation", "Comedy", "Family", "Foreign", "Sci-Fi", "Travel", "Children",
      "Drama", "Horror", "Action", "Classics", "Games", "New", "Documentary", "Sports", "Music"));

-- Valid Special Feature Constraint
alter table film
   add constraint feature_check check (special_features in ("Behind the Scenes", "Commentaries", "Deleted Scenes", "Trailers"));

-- Valid Date Constraints
alter table film add constraint release_year_check check (release_year between 1850 and 2025);
alter table rental
   modify column rental_date datetime,
   modify column return_date datetime,
   add constraint rent_return_check check (return_date >= rental_date);
alter table payment modify column payment_date datetime;

-- Valid Activity State Constraints
alter table staff add constraint staff_activity_check check (active in (0, 1));
alter table customer add constraint customer_activity_check check (active in (0, 1));
-- Valid Rental Durations Constraint
alter table film add constraint duration_check check (rental_duration between 2 and 8);

-- Valid Daily Rental Rate Constraint
alter table film add constraint rate_check check (rental_rate between 0.99 and 6.99);

-- Valid Movie Length Constraint
alter table film add constraint length_check check (length between 30 and 200);

-- Valid Movie Rating Constraint
alter table film add constraint rating_check check (rating in ("PG", "G", "NC-17", "PG-13", "R"));

-- Valid Movie Replacement Cost Constraint
alter table film add constraint replacement_check check (replacement_cost between 5.00 and 100.00);

-- Valid Movie Payment Amount Constraint
alter table payment add constraint payment_check check (amount >= 0);
```

1. **What is the average length of films in each category? List the results in alphabetical order of categories.**

```
-- #1 Displays length of movies for each category rounded to the nearest whole number
select round(avg(length), 0) as avg_length, category.name
from film join film_category using (film_id)
join category using (category_id)
group by category.name;
```

This query displays a list of all the average movie lengths grouped by movie category. I joined the tables film, film category, and category to connect the data and used keywords avg and round to get the average and round it to the nearest whole.

avg_length	name
112	Action
111	Animation
110	Children
112	Classics
116	Comedy
109	Documentary
121	Drama
115	Family
122	Foreign
128	Games
112	Horror
114	Music
111	New
108	Sci-Fi
128	Sports
113	Travel

2. **Which categories have the longest and shortest average film lengths?**

```
-- #2a Displays the longest average category of movies
 select round(avg(length), 0) as avg_length, category.name
from film join film_category using (film_id)
join category using (category_id)
group by category.name
order by avg_length desc
limit 1;
```

```
-- #2b Displays the shortest average category of movies
 select round(avg(length), 0) as avg_length, category.name
from film join film_category using (film_id)
join category using (category_id)
group by category.name
order by avg_length asc
limit 1;
```

These queries display both the single longest and shortest average category movie. These queries are designed almost identically to the last query. The difference is that they are ordered by average movie length and limited by 1. The first one is

ordered descending to display the longest average, and the other is ordered ascending to display the shortest average.

	avg_length	name
▶	128	Games

	avg_length	name
▶	108	Sci-Fi

3. Which customers have rented action but not comedy or classic movies?

```
-- #3 Displays all customers who have rented an action movie but not comedy or classic, using a view and subquery
create view customer_categories as
   select distinct customer_id, customer.first_name, customer.last_name, category.name as category_name
   from customer join rental using (customer_id)
   join inventory using (inventory_id)
   join film_category using (film_id)
   join category using (category_id);

select customer_id, first_name, last_name, category_name
from customer_categories c1
where category_name = "Action" and customer_id not  in (
   select customer_id
   from customer_categories c2
   where c2.category_name in ("Comedy", "Classics"));
```

For this query, I created a view of the needed tables to reduce the joins needed for the query and subquery. I then printed a table of the names of customers who rented movies in the category action, and using a subquery and the keyword not in, I excluded any customer who rented a movie in the comedy or classics category.

	customer_id	first_name	last_name	category_name
▶	361	LAWRENCE	LAWTON	Action
	323	MATTHEW	MAHAN	Action
	452	TOM	MILNER	Action
	250	JO	FOWLER	Action
	330	SCOTT	SHELLEY	Action
	432	EDWIN	BURK	Action
	164	JOANN	GARDNER	Action
	17	DONNA	THOMPSON	Action
	433	DON	BONE	Action
	350	JUAN	FRALEY	Action
	171	DOLORES	WAGNER	Action
	445	MICHEAL	FORMAN	Action
	139	AMBER	DIXON	Action

4. **Which actor has appeared in the most English-language movies?**

```
-- #4 Displays the actor found in the most amount of english spoken movies
select count(film_id) as total_english_films, actor.first_name, actor.last_name
from actor join film_actor using(actor_id)
join film using (film_id)
join language using (language_id)
group by language.name, actor_id
having language.name = "English"
order by total_english_films desc
limit 1;
```

This query prints out one actor who has appeared the most in English movies. I did this by joining actor, film_actor, film, and language. Then I grouped by language name, having the language name be equal to English. Then I ordered the aggregate, film_id count, to have the largest appearance on top, and limited it by one.

	total_english_films	first_name	last_name
▶	42	GINA	DEGENERES

5. **How many distinct movies were rented for exactly 10 days from the store where Mike works?**

```
-- #5 Displays the number of distanct movies rented for 10 days from the store that Mike works at
select count(distinct(film_id)) as movies_rented_for_10_days
from film f join inventory using (film_id)
join rental r using (inventory_id)
join staff sta using (staff_id)
join store sto on sta.store_id = sto.store_id
where datediff(return_date, rental_date) = 10 and sto.store_id = (
  select sto2.store_id
  from store sto2 join staff sta2 on sto2.store_id = sta2.store_id
  where sta2.first_name = "Mike");
```

This query prints the total number of distinct movies that were rented for ten days from the store where Mike works. I did this by joining the necessary tables and using the datediff function to determine the ten-day rental window and a subquery to make sure it checks only Mike's store.

	movies_rented_for_10_days
▶	47

6. **Alphabetically list actors who appeared in the movie with the largest cast of actors.**

```
-- #6 Displays the names of cast members that starred in the movie with the largest cast in an alphabetical order
select a.last_name, a.first_name, film.title
from film join film_actor using (film_id)
join actor a using (actor_id)
where film_id = (
   select film_id
   from film join film_actor using (film_id)
   group by film_id
   order by count(actor_id) desc
   limit 1)
order by a.last_name, a.first_name;
```

This query prints the full name of all actors who appeared in the largest cast movie. I found the largest cast movie by using a subquery grouped by the film id and ordered by the actor count, limiting one, to have the one largest on top. I used a where condition to print only actors from that movie and ordered by last and first name so the names are printed alphabetically.

last_name	first_name	title
BACALL	RUSSELL	LAMBS CINCINATTI
BALL	RENEE	LAMBS CINCINATTI
BARRYMORE	JULIA	LAMBS CINCINATTI
BOLGER	VAL	LAMBS CINCINATTI
DAMON	SCARLETT	LAMBS CINCINATTI
DEE	EMILY	LAMBS CINCINATTI
GOODING	EWAN	LAMBS CINCINATTI
HESTON	GEOFFREY	LAMBS CINCINATTI
HOFFMAN	WOODY	LAMBS CINCINATTI
KILMER	REESE	LAMBS CINCINATTI
NEESON	CHRISTIAN	LAMBS CINCINATTI
PFEIFFER	OLYMPIA	LAMBS CINCINATTI
POSEY	BURT	LAMBS CINCINATTI
TEMPLE	MENA	LAMBS CINCINATTI
TORN	WALTER	LAMBS CINCINATTI