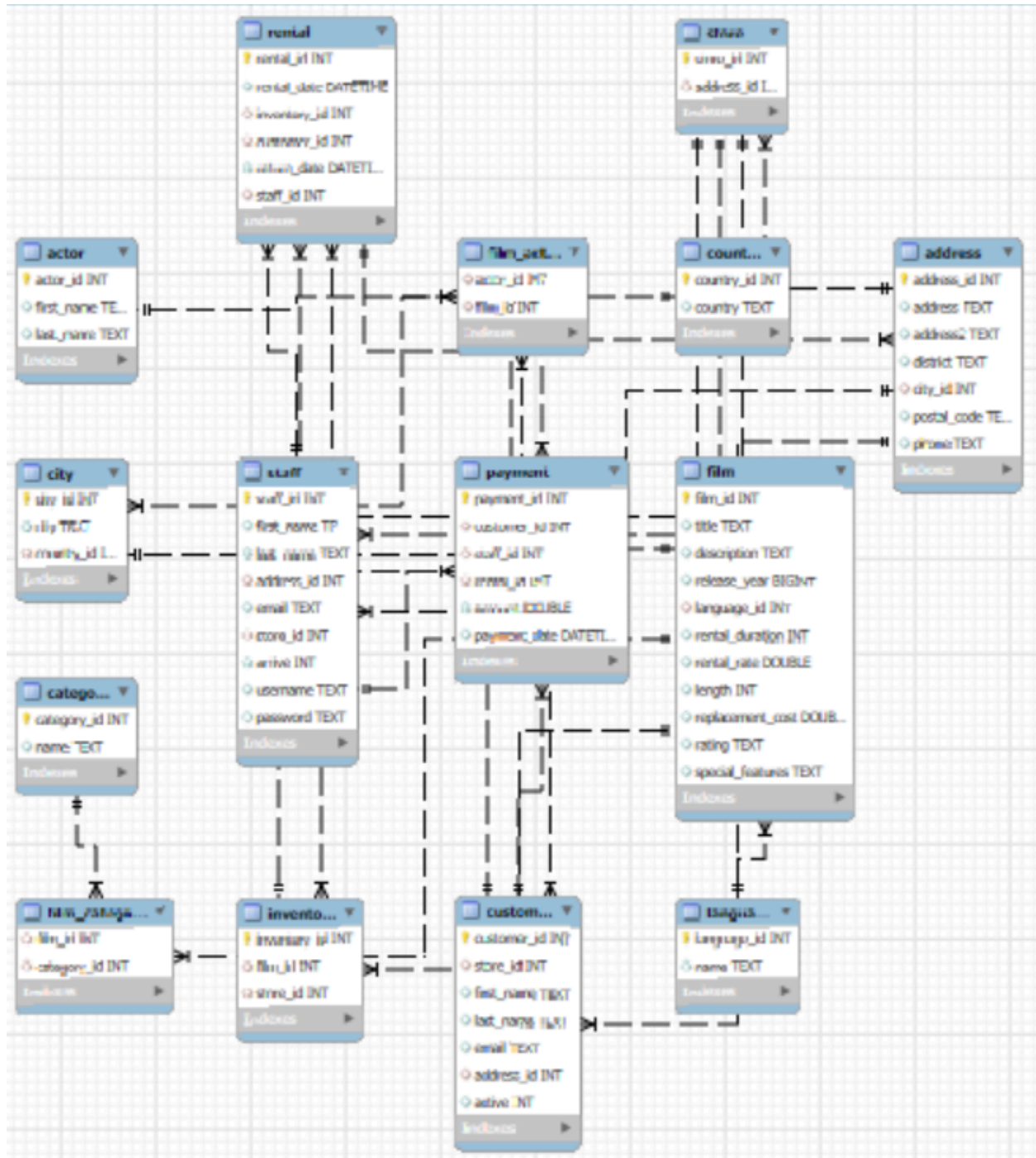Title: Database Assignment 4
Your Name: Jack Saunders
Date: 11/4/2024

ER Diagram

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

This query shows the name and average length of each category of film since it's grouped by category name	**Answer:**

```
58    -- 1) What is the average length of films in each category? List the results in alphabetic order o
59 •  select name, avg(length)
60    from category join film_category on category.category_id = film_category.category_id
61    join film on film_category.film_id = film.film_id
62    group by name;
```

name	avg(length)
Action	111.6094
Animation	111.0152
Children	109.8000
Classics	111.6667
Comedy	115.8276
Documentary	108.7500
Drama	120.8387
Family	114.7826

Result 236 ×

Output

Action Output

#	Time	Action	Message
630	23:01:54	select name, avg(length) from film join film_category on film.film_id = film_category.film_id join category on film_category.category_id = category_j...	1 row(s) returned
631	23:05:39	select name, avg(length) from category join film_category on category.category_id = film_category.category_id join film on film_category.film_id = film.film_id...	16 row(s) returned

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

Lines 91 through 98 shows the name and average film length. Similar to query 1, lines 92 through 96 find the average length of each film category. This is saved as a view in line 92, limited to one answer in line 98 and ordered in from smallest to largest average film time. The same occurs with lines 100 through 107 just from biggest to smallest according to line 106. Lastly, line 109 shows both saved views.	**Answer:**

```
91 •  create view low as
92    select name, avg(length)
93        from film
94        join film_category on film.film_id = film_category.film_id
95        join category on film_category.category_id = category.category_id
96        group by name
97        order by avg(length)
98        limit 1;
99
100 • create view high as
101    select name, avg(length)
102        from film
103        join film_category on film.film_id = film_category.film_id
104        join category on film_category.category_id = category.category_id
105        group by name
106        order by avg(length) desc
107        limit 1;
108
109 • select * from low join high; -- answer
110
```

Result Grid   Filter Rows: Export: Wrap Cell Content:

name	avg(length)	name	avg(length)
Sci-Fi	108.1967	Sports	128.2027

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

Lines 112 through 118 finds all the	**Answer:**

unique customer last names that have rented action movies. Lines 120 through 126 does the same thing but for classics and comedy movies. The except clause at line 119 is a visual bug, but subtracts the second half of the query from the first; only customers who have rented action films and have not rented classics nor comedy films are returned.

```
111    -- 3) Which customers have rented action but not comedy or classic movies?
112 •  select distinct(last_name)
113    from customer
114    join rental using (customer_id)
115    join inventory using (inventory_id)
116    join film_category using (film_id)
117    join category using (category_id)
118    where name = 'action'
119 ⊠  except
120    select distinct(last_name)
121    from customer
122    join rental using (customer_id)
123    join inventory using (inventory_id)
124    join film_category using (film_id)
125    join category using (category_id)
126    where name = 'classics' or name = 'comedy'
127    order by last_name asc;
128
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: ⓘ

last_name
▶ BONE
BURK
DIXON
FERNANDEZ
FORMAN
FOWLER
FRALEY
GARDNER

Result 238 ×

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓	632	23:18:36	select * from low join high LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.00
✓	633	23:24:57	select distinct(last_name) from customer join rental usin...	17 row(s) returned	0.063 sec / 0.00

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

Line 130 finds the actor's first and last name and counts how many films they were in. The list is in order of how many english films they were in. It is limited to only english movies in line 139 and 133.

Answer:

```
129    -- 4) Which actor has appeared in the most English-language
130 •  select distinct first_name, last_name, count(*) from actor
131    join film_actor using (actor_id)
132    join film using (film_id)
133    where language_id = 1
134    group by actor_id
135    having count(actor_id) >= All
136 ⊖     (select first_name from actor
137        join film_actor using (actor_id)
138        join film using (film_id)
139        where language_id = 1
140        group by film_id)
141    order by count(*) desc
142    limit 1;
143
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: ⓘ | Fetch rows:

first_name	last_name	count(*)
▶ GINA	DEGENERES	42

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

Line 144 counts the distinct number of films. Line 149 limits that answer by using datediff() on the dates where the movie was taken and returned. The count of the movies that were rented for only ten days were given and further limited by tuples which had Mike as a staff at the store. Only 61 tuples fit all criteria

Answer:

```
144 •  select count(distinct film_id)
145    from rental
146    join inventory using (inventory_id)
147    join film using (film_id)
148    join staff using (store_id)
149    where datediff(return_date, rental_date) = 10 and first_name = 'Mike';
150
151    -- 6) Alphabetically list actors who appeared in the movie with the lar
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: IA

count(distinct film_id)
61

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

Line 152 finds the actor's first and last name and counts how many films they were in. The list is in order of first name.

Answer:

```
152 •  select distinct first_name, last_name, count(actor_id) from actor
153    join film_actor using (actor_id)
154    group by actor_id
155    order by first_name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: IA

first_name	last_name	count(actor_id)
ADAM	GRANT	18
ADAM	HOPPER	22
AL	GARLAND	26
ALAN	DREYFUSS	27
ALBERT	JOHANSSON	33
ALBERT	NOLTE	31
ALEC	WAYNE	29
ANGELA	HUDSON	34

Result 252 ×