DB Assignment 4

Giannina Flamiano

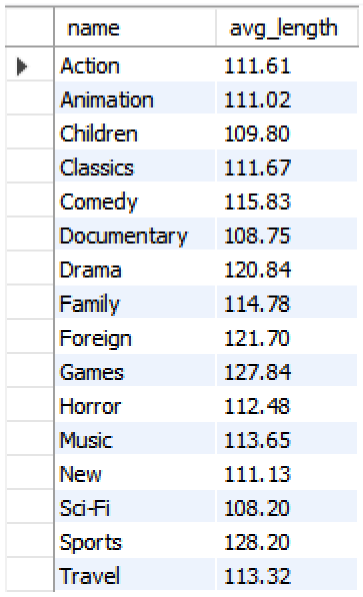
October 31, 2024

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

SQL QUERY

```
SELECT c.name, ROUND(AVG(f.length), 2) AS avg_length
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
GROUP BY c.name
ORDER BY c.name;
```

SCREENSHOT



name	avg_length
Action	111.61
Animation	111.02
Children	109.80
Classics	111.67
Comedy	115.83
Documentary	108.75
Drama	120.84
Family	114.78
Foreign	121.70
Games	127.84
Horror	112.48
Music	113.65
New	111.13
Sci-Fi	108.20
Sports	128.20
Travel	113.32

EXPLANATION

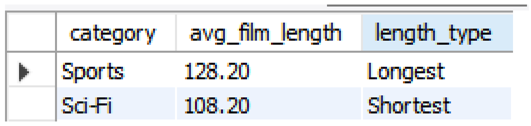
This query looks at film, film_category, and category tables. It first joins the film_category table to get the film_category details based on matching film IDs. Then, it joins the category table to get the category details based on matching category IDs. In the results table, the query will select the category name and the average film length. The results are grouped by category name and are sorted in alphabetical order of categories.

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

SQL QUERY

```
WITH MaxFilmLength AS (
      SELECT c.name AS category, ROUND(AVG(f.length), 2) AS avg_film_length, 'Longest' AS
            length_type
      FROM film f
      JOIN film_category fc ON f.film_id = fc.film_id
      JOIN category c ON fc.category_id = c.category_id
      GROUP BY c.name
      ORDER BY avg_film_length DESC
      LIMIT 1
), MinFilmLength AS (
      SELECT c.name AS category, ROUND(AVG(f.length), 2) AS avg_film_length, 'Shortest' AS
            length_type
      FROM film f
      JOIN film_category fc ON f.film_id = fc.film_id
      JOIN category c ON fc.category_id = c.category_id
      GROUP BY c.name
      ORDER BY avg_film_length
      LIMIT 1
) SELECT * FROM MaxFilmLength UNION SELECT * FROM MinFilmLength;
```

SCREENSHOT



category	avg_film_length	length_type
Sports	128.20	Longest
Sci-Fi	108.20	Shortest

EXPLANATION

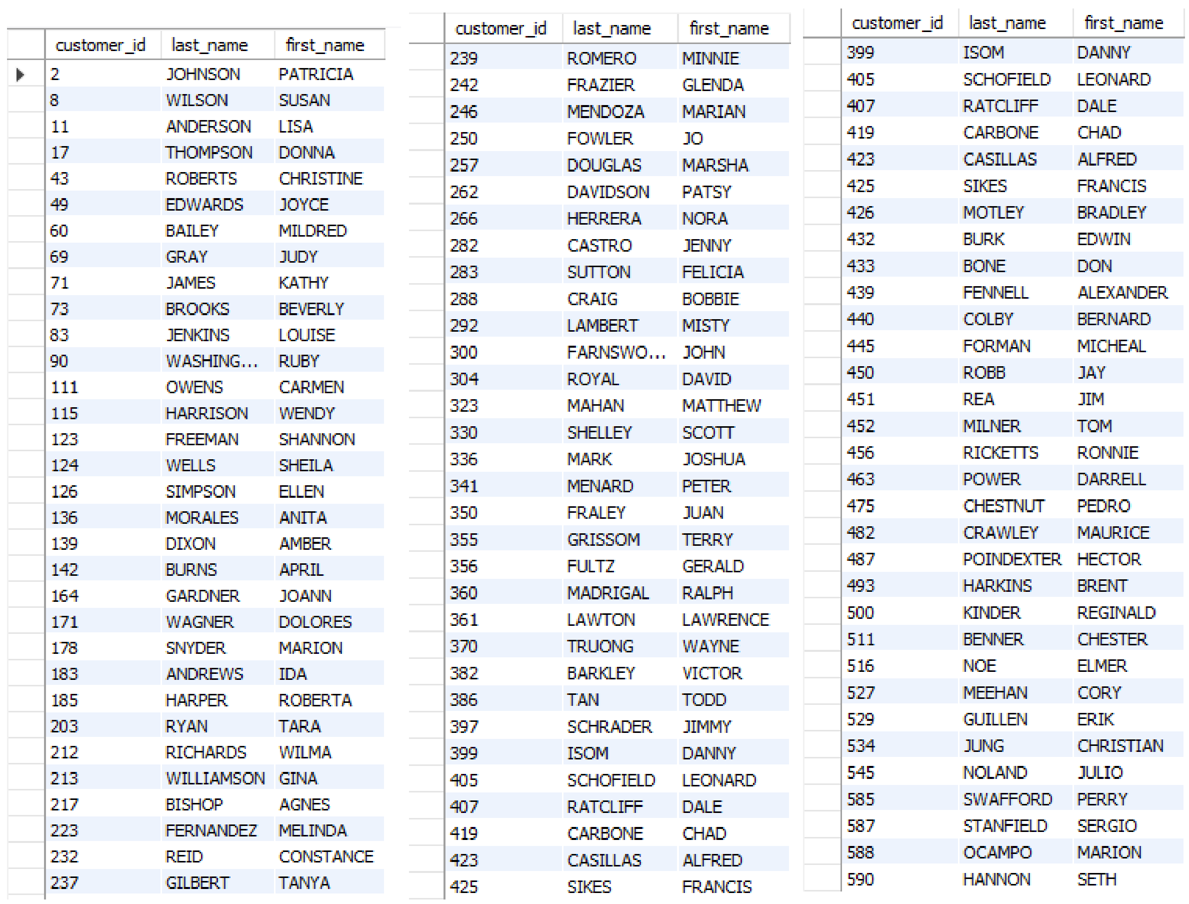
This query has two CTE's: MaxFilmLength and MinFilmLength. Both look at film, film_category, and category tables. They both first join the film_category table to get the film_category details based on matching film IDs. Then, it joins the category table to get the category details based on matching category IDs. In the results table for both CTEs, the query will select the category name, the average film length rounded to two decimal places, and length type. Length type will vary based on CTE. In MaxFilmLength, the length type will be 'Longest' and the results are grouped by category name and sorted by the average film length in descending order. The results are then limited by 1 so that we get the longest average length film. On the contrary, in MinFilmLength, the length type will be 'Shortest'. The results are also grouped by category name, but are sorted by the average film length in ascending order. The results are limited by 1 so that we get the shortest average film length. In the main query, the results table will union two select all queries from the respective CTEs.

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

SQL QUERY

```
WITH CustomersAction AS (
	SELECT r.customer_id
	FROM rental r
	JOIN inventory i ON r.inventory_id = i.inventory_id
	JOIN film_category fc ON i.film_id = fc.film_id
	JOIN category c ON fc.category_id = c.category_id
	WHERE c.name = 'Action'
	GROUP BY r.customer_id
), CustomersComedyClassic AS (
	SELECT r.customer_id
	FROM rental r
	JOIN inventory i ON r.inventory_id = i.inventory_id
	JOIN film_category fc ON i.film_id = fc.film_id
	JOIN category c ON fc.category_id = c.category_id
	WHERE c.name = 'Comedy' OR c.name = 'Classic'
	GROUP BY r.customer_id
)
SELECT ca.customer_id, c.last_name, c.first_name
FROM CustomersAction ca
JOIN customer c ON ca.customer_id = c.customer_id
WHERE ca.customer_id NOT IN (SELECT * FROM CustomersComedyClassic)
ORDER BY ca.customer_id;
```

SCREENSHOT



customer_id	last_name	first_name
2	JOHNSON	PATRICIA
8	WILSON	SUSAN
11	ANDERSON	LISA
17	THOMPSON	DONNA
43	ROBERTS	CHRISTINE
49	EDWARDS	JOYCE
60	BAILEY	MILDRED
69	GRAY	JUDY
71	JAMES	KATHY
73	BROOKS	BEVERLY
83	JENKINS	LOUISE
90	WASHING...	RUBY
111	OWENS	CARMEN
115	HARRISON	WENDY
123	FREEMAN	SHANNON
124	WELLS	SHEILA
126	SIMPSON	ELLEN
136	MORALES	ANITA
139	DIXON	AMBER
142	BURNS	APRIL
164	GARDNER	JOANN
171	WAGNER	DOLORES
178	SNYDER	MARION
183	ANDREWS	IDA
185	HARPER	ROBERTA
203	RYAN	TARA
212	RICHARDS	WILMA
213	WILLIAMSON	GINA
217	BISHOP	AGNES
223	FERNANDEZ	MELINDA
232	REID	CONSTANCE
237	GILBERT	TANYA

customer_id	last_name	first_name
239	ROMERO	MINNIE
242	FRAZIER	GLENDA
246	MENDOZA	MARIAN
250	FOWLER	JO
257	DOUGLAS	MARSHA
262	DAVIDSON	PATSY
266	HERRERA	NORA
282	CASTRO	JENNY
283	SUTTON	FELICIA
288	CRAIG	BOBBIE
292	LAMBERT	MISTY
300	FARNSWO...	JOHN
304	ROYAL	DAVID
323	MAHAN	MATTHEW
330	SHELLEY	SCOTT
336	MARK	JOSHUA
341	MENARD	PETER
350	FRALEY	JUAN
355	GRISSOM	TERRY
356	FULTZ	GERALD
360	MADRIGAL	RALPH
361	LAWTON	LAWRENCE
370	TRUONG	WAYNE
382	BARKLEY	VICTOR
386	TAN	TODD
397	SCHRADER	JIMMY
399	ISOM	DANNY
405	SCHOFIELD	LEONARD
407	RATCLIFF	DALE
419	CARBONE	CHAD
423	CASILLAS	ALFRED
425	SIKES	FRANCIS

customer_id	last_name	first_name
399	ISOM	DANNY
405	SCHOFIELD	LEONARD
407	RATCLIFF	DALE
419	CARBONE	CHAD
423	CASILLAS	ALFRED
425	SIKES	FRANCIS
426	MOTLEY	BRADLEY
432	BURK	EDWIN
433	BONE	DON
439	FENNELL	ALEXANDER
440	COLBY	BERNARD
445	FORMAN	MICHEAL
450	ROBB	JAY
451	REA	JIM
452	MILNER	TOM
456	RICKETTS	RONNIE
463	POWER	DARRELL
475	CHESTNUT	PEDRO
482	CRAWLEY	MAURICE
487	POINDEXTER	HECTOR
493	HARKINS	BRENT
500	KINDER	REGINALD
511	BENNER	CHESTER
516	NOE	ELMER
527	MEEHAN	CORY
529	GUILLEN	ERIK
534	JUNG	CHRISTIAN
545	NOLAND	JULIO
585	SWAFFORD	PERRY
587	STANFIELD	SERGIO
588	OCAMPO	MARION
590	HANNON	SETH

EXPLANATION

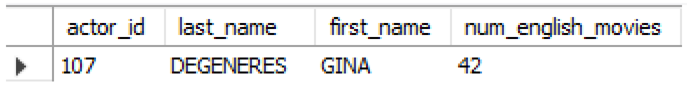
This query has two CTEs: CustomersAction and CustomersComedyClassic. Both look at rental, inventory, film_category, and category tables. They both first join the rental table to get the rental details based on matching inventory IDs. Then, they join the film_category table to get the film_category details based on matching film IDs. Then, they join the category table to get the category details based on matching category IDs. CustomersAction looks for the ‘Action’ category name and groups the results by customer ID. The results for this CTE is then a list of customer IDs who have rented action movies. CustomersComedyClassic looks for the ‘Comedy’ or ‘Classic’ category names and groups the results by customer ID. The results for this CTE is then a list of customer IDs who have rented comedy or classic movies. The main query looks at the two CTEs and the customer table. The query first joins the customer table to get the customer details based on matching customer IDs. Then it looks for customer IDs that are in CustomersAction but not in CustomersComedyClassic and groups the results by customer ID. The results table will display the customer’s ID, first name, and last name of those who have rented action but not comedy or classic movies.

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

SQL QUERY

```
WITH EnglishMovies AS (
	SELECT f.film_id
	FROM film f
	JOIN language l ON f.language_id = l.language_id
	WHERE l.name = 'English'
)
SELECT fa.actor_id, a.last_name, a.first_name, COUNT(fa.film_id) AS num_english_movies
FROM film_actor fa
JOIN actor a ON fa.actor_id = a.actor_id
WHERE film_id IN (SELECT * FROM EnglishMovies)
GROUP BY actor_id
ORDER BY num_english_movies DESC LIMIT 1;
```

SCREENSHOT



actor_id	last_name	first_name	num_english_movies
107	DEGENERES	GINA	42

EXPLANATION

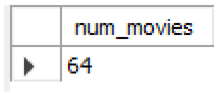
This query has one CTE: EnglishMovies. It looks at film and language tables. It first joins the language table to get language details based on matching language IDs. Then it looks for films that have English as the language name. The results return a list of film IDs that are in English. The main query looks at film_actor and actor tables and the CTE created. It first joins the actor table to get the actor details based on matching actor IDs. Then it looks for the film IDs that are in the CTE, EnglishMovies. In the results table, the query will select the actor's ID, last name, first name, and the number of English-language movies s/he has appeared in. The results are grouped by actor ID and sorted by the number of movies in descending order. The results are limited by 1 so that we get the actor who has appeared in the most English-language movies.

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

SQL QUERY

```
SELECT DISTINCT COUNT(i.film_id) AS num_movies
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
WHERE DATEDIFF(r.return_date, r.rental_date) = 10
AND i.store_id IN (SELECT store_id FROM staff WHERE first_name = 'Mike');
```

SCREENSHOT



	num_movies
▶	64

EXPLANATION

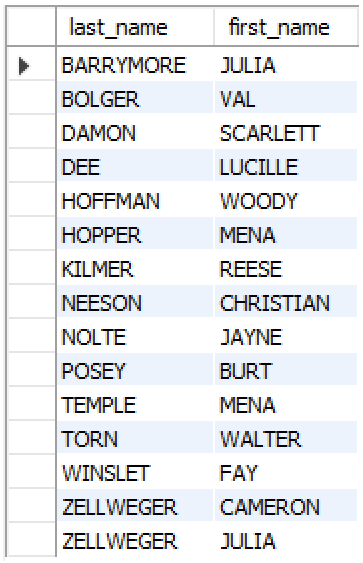
This query looks at rental and inventory tables. It first joins the inventory table to get the inventory details based on matching inventory IDs. Then it looks for films that were rented for 10 days using a DATEDIFF() function and have a store ID that's in the results table of a subquery. The subquery looks at the staff table and returns the list of store IDs where Mike works. The results table of the main query returns the distinct count of movies that were rented for 10 days from the store where Mike works.

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

SQL QUERY

```
WITH LargestCast AS (
	SELECT film_id, COUNT(actor_id) AS cast_num
	FROM film_actor
	GROUP BY film_id
	ORDER BY cast_num DESC LIMIT 1
)
SELECT a.last_name, a.first_name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
WHERE fa.film_id IN (SELECT film_id FROM LargestCast)
ORDER BY a.last_name;
```

SCREENSHOT

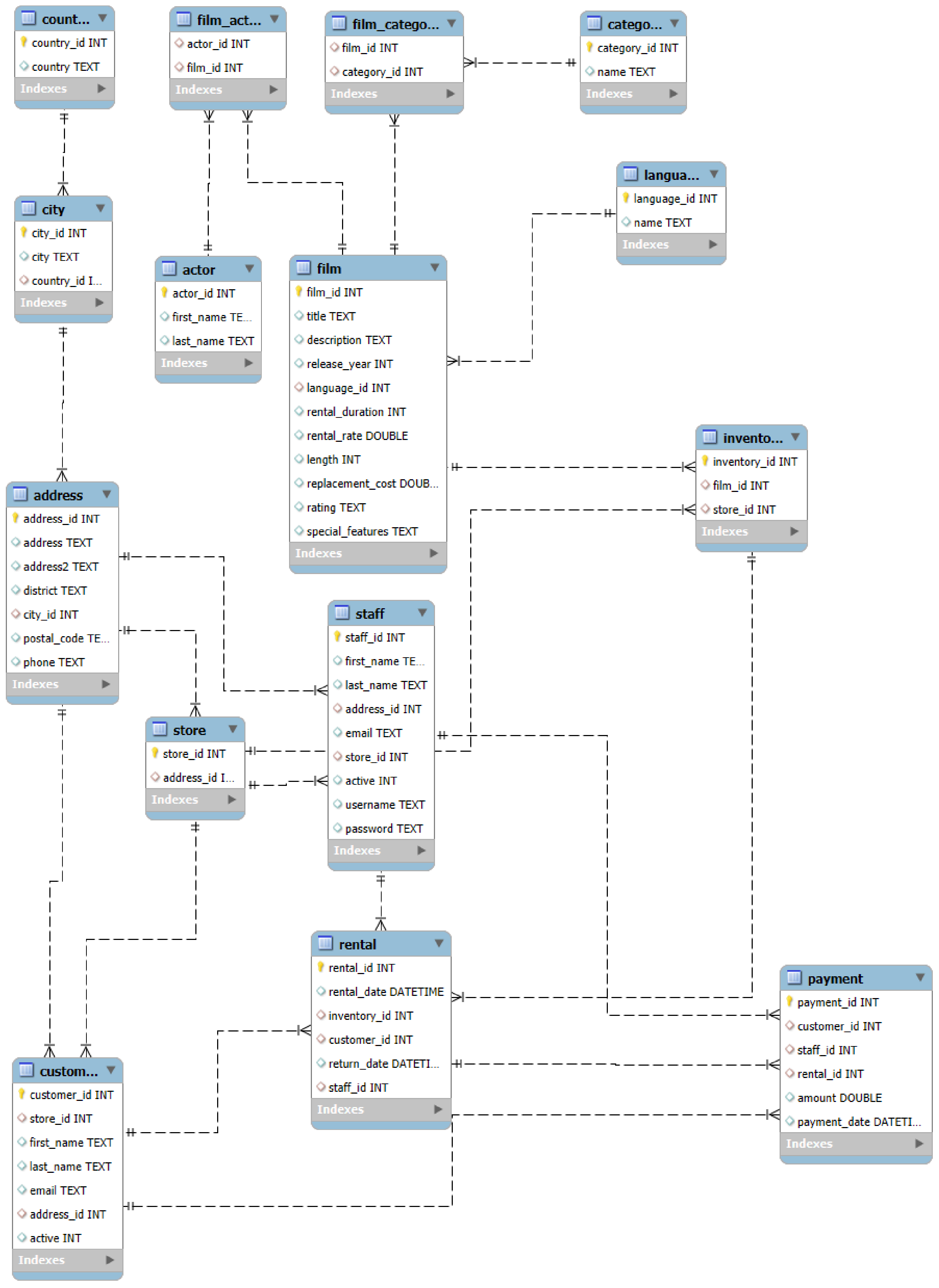


last_name	first_name
BARRYMORE	JULIA
BOLGER	VAL
DAMON	SCARLETT
DEE	LUCILLE
HOFFMAN	WOODY
HOPPER	MENA
KILMER	REESE
NEESON	CHRISTIAN
NOLTE	JAYNE
POSEY	BURT
TEMPLE	MENA
TORN	WALTER
WINSLET	FAY
ZELLWEGER	CAMERON
ZELLWEGER	JULIA

EXPLANATION

This query has one CTE: LargestCast, and it looks at the film_actor table. In the results table, the query will select the film ID and the count of actor IDs with an alias cast_num. The results are grouped by film ID and ordered by cast number in descending order with a limit 1, so that we get the movie with the largest cast. The main query looks at actor and film_actor tables and the CTE created. The query first joins the film_actor table to get the film_actor details based on matching actor IDs. Then it looks for film ID matches with the film ID for the movie with the largest cast, which we get from LargestCast. In the results table, the actor's last and first name will be displayed. The results are sorted in ascending order by last name, which gives us the list of actors who appeared in the movie with the largest cast of actors in alphabetical order.

ERD Diagram


count...
country_id INT
country TEXT
Indexes
film_act...
actor_id INT
film_id INT
Indexes
film_catego...
film_id INT
category_id INT
Indexes
catego...
category_id INT
name TEXT
Indexes
langua...
language_id INT
name TEXT
Indexes
city
city_id INT
city TEXT
country_id I...
Indexes
actor
actor_id INT
first_name TE...
last_name TEXT
Indexes
film
film_id INT
title TEXT
description TEXT
release_year INT
language_id INT
rental_duration INT
rental_rate DOUBLE
length INT
replacement_cost DOUB...
rating TEXT
special_features TEXT
Indexes
invento...
inventory_id INT
film_id INT
store_id INT
Indexes
address
address_id INT
address TEXT
address2 TEXT
district TEXT
city_id INT
postal_code TE...
phone TEXT
Indexes
staff
staff_id INT
first_name TE...
last_name TEXT
address_id INT
email TEXT
store_id INT
active INT
username TEXT
password TEXT
Indexes
store
store_id INT
address_id I...
Indexes
rental
rental_id INT
rental_date DATETIME
inventory_id INT
customer_id INT
return_date DATETI...
staff_id INT
Indexes
payment
payment_id INT
customer_id INT
staff_id INT
rental_id INT
amount DOUBLE
payment_date DATETI...
Indexes
custom...
customer_id INT
store_id INT
first_name TEXT
last_name TEXT
email TEXT
address_id INT
active INT
Indexes