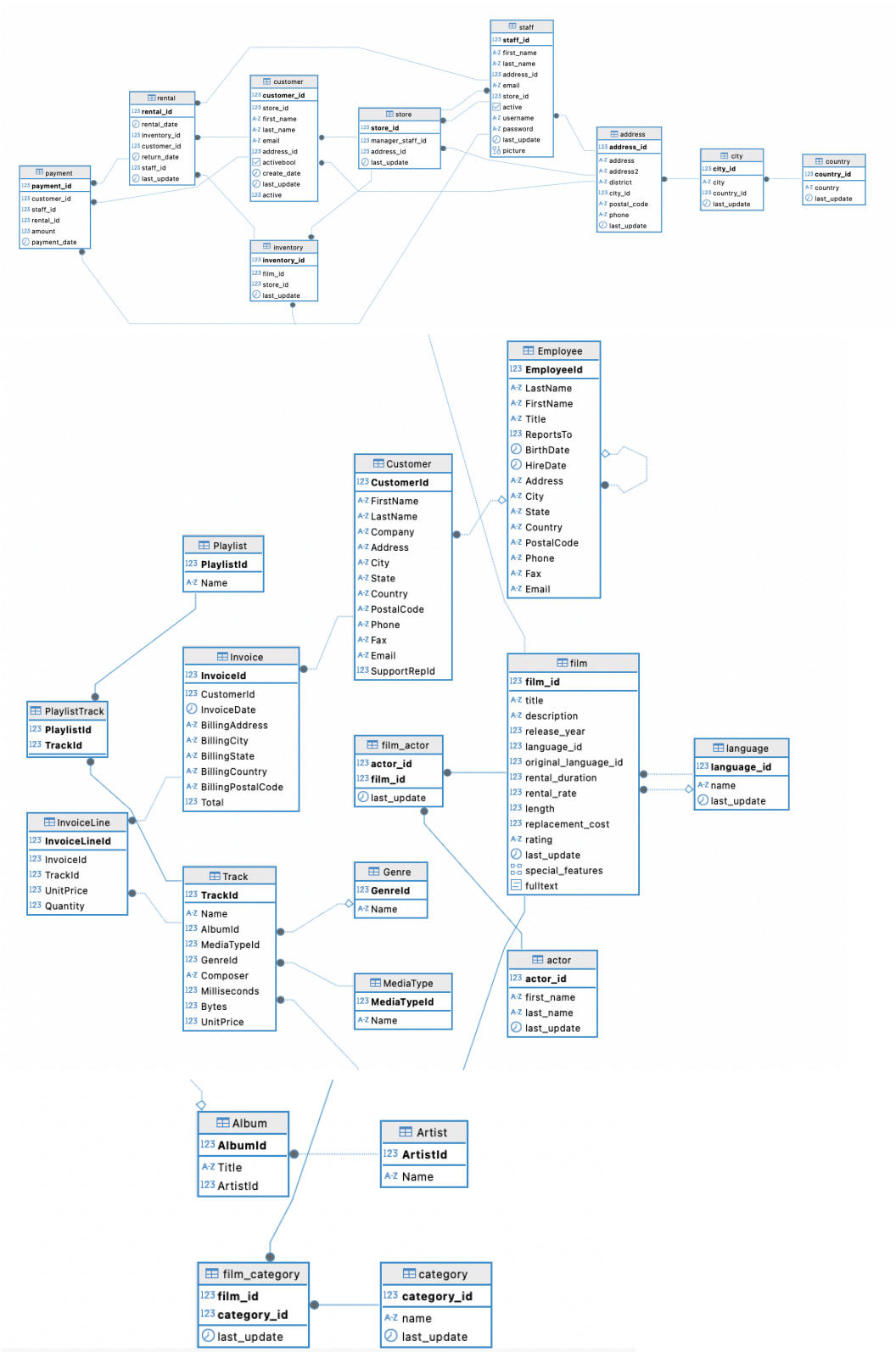
DataProject: LógicaConsultasSQL

1. Crea el esquema de la BBDD.



2. Muestra los nombres de todas las películas con una clasificación por edades de ‘R’.

```
SELECT f.title
FROM film f
WHERE rating = 'R';
```

3. Encuentra los nombres de los actores que tengan un “actor_id” entre 30 y 40.

```
SELECT first_name, last_name
FROM actor
WHERE actor_id BETWEEN 30 AND 40;
```

4. Obtén las películas cuyo idioma coincide con el idioma original.

```
SELECT title
FROM film
WHERE language_id = original_language_id;
```

5. Ordena las películas por duración de forma ascendente.

```
SELECT *
FROM film
ORDER BY length ASC;
```

6. Encuentra el nombre y apellido de los actores que tengan ‘Allen’ en su apellido.

```
SELECT first_name, last_name
FROM actor
WHERE last_name = 'ALLEN';
```

7. Encuentra la cantidad total de películas en cada clasificación de la tabla "film" y muestra la clasificación junto con el recuento.

```
SELECT c.name AS category_name, COUNT(*) AS total_movies
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
GROUP BY c.name;
```

8. Encuentra el título de todas las películas que son 'PG-13' o tienen una duración mayor a 3 horas en la tabla film.

```
SELECT title
FROM film
WHERE rating = 'PG-13' OR rental_duration > 180;
```

9. Encuentra la variabilidad de lo que costaría reemplazar las películas.

```
SELECT STDDEV(replacement_cost) AS variabilidad_reemplazo
FROM film;
```

10. Encuentra la mayor y menor duración de una película de nuestra BBDD.

```
SELECT MAX(length) AS mayor_duracion, MIN(length) AS menor_duracion
FROM film;
```

11. Encuentra lo que costó el antepenúltimo alquiler ordenado por día.

```
WITH RankedPayments AS (
  SELECT *, ROW_NUMBER() OVER (ORDER BY payment_date DESC) AS rn
  FROM payment
)
SELECT amount
FROM RankedPayments
WHERE rn = 2;
```

12. Encuentra el título de las películas en la tabla “film” que no sean ni ‘NC 17’ ni ‘G’ en cuanto a su clasificación.

```sql
SELECT title
FROM film
WHERE rating NOT IN ('NC-17', 'G');
```

13. Encuentra el promedio de duración de las películas para cada clasificación de la tabla film y muestra la clasificación junto con el promedio de duración.

```sql
SELECT rating, AVG(length) AS promedio_duracion
FROM film
GROUP BY rating;
```

14. Encuentra el título de todas las películas que tengan una duración mayor a 180 minutos.

```sql
SELECT title
FROM film
WHERE length > 180;
```

15. ¿Cuánto dinero ha generado en total la empresa?

```sql
SELECT SUM(amount) AS total_ingresos
FROM payment;
```

16. Muestra los 10 clientes con mayor valor de id.

```sql
SELECT *
FROM customer
ORDER BY customer_id DESC
LIMIT 10;
```

17. Encuentra el nombre y apellido de los actores que aparecen en la película con título ‘Egg Igby’.

```
SELECT a.first_name, a.last_name
FROM film f
JOIN film_actor fa ON f.film_id = fa.film_id
JOIN actor a ON fa.actor_id = a.actor_id
WHERE f.title = 'EGG IGBY';
```

18. Selecciona todos los nombres de las películas únicas.

```
SELECT DISTINCT title
FROM film;
```

19. Encuentra el título de las películas que son comedias y tienen una duración mayor a 180 minutos en la tabla “film”.

```
SELECT f.title
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Comedy' AND f.length > 180;
```

20. Encuentra las categorías de películas que tienen un promedio de duración superior a 110 minutos y muestra el nombre de la categoría junto con el promedio de duración.

```
SELECT c.name AS category_name, AVG(f.length) AS promedio_duracion
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
GROUP BY c.name
HAVING AVG(f.length) > 110;
```

21. ¿Cuál es la media de duración del alquiler de las películas?

```
SELECT AVG(length) AS media_duracion_alquiler
FROM film;
```

22. Crea una columna con el nombre y apellidos de todos los actores y actrices.

```
SELECT CONCAT(first_name, ' ', last_name) AS nombre_completo
FROM actor;
```

23. Números de alquiler por día, ordenados por cantidad de alquiler de forma descendente.

```
SELECT payment_date, COUNT(*) AS cantidad_alquileres
FROM payment
GROUP BY payment_date
ORDER BY cantidad_alquileres DESC;
```

24. Encuentra las películas con una duración superior al promedio.

```
SELECT *
FROM film
WHERE length > (SELECT AVG(length) FROM film);
```

25. Averigua el número de alquileres registrados por mes.

```
SELECT
  EXTRACT(YEAR FROM payment_date) AS año,
  EXTRACT(MONTH FROM payment_date) AS mes,
  COUNT(*) AS total_alquileres
FROM payment
GROUP BY año, mes
ORDER BY año, mes;
```

26. Encuentra el promedio, la desviación estándar y varianza del total pagado.

```
SELECT
  AVG(amount) AS promedio,
  STDDEV(amount) AS desviacion_estandar,
  VARIANCE(amount) AS varianza
FROM payment;
```

27. ¿Qué películas se alquilan por encima del precio medio?

```
SELECT title, rental_rate
FROM film
WHERE rental_rate > (SELECT AVG(rental_rate) FROM film);
```

28. Muestra el id de los actores que hayan participado en más de 40 películas.

```
SELECT actor_id
FROM film_actor
GROUP BY actor_id
HAVING COUNT(film_id) > 40;
```

29. Obtener todas las películas y, si están disponibles en el inventario, mostrar la cantidad disponible.

```
SELECT
  f.film_id,
  f.title,
  COUNT(i.inventory_id) AS cantidad_disponible
FROM
  film f
LEFT JOIN
  inventory i ON f.film_id = i.film_id
GROUP BY
  f.film_id, f.title;
```

30. Obtener los actores y el número de películas en las que ha actuado.

```
SELECT a.actor_id, a.first_name, a.last_name, COUNT(fa.film_id) AS numero_peliculas
FROM actor a
LEFT JOIN film_actor fa ON a.actor_id = fa.actor_id
GROUP BY a.actor_id, a.first_name, a.last_name;
```

31. Obtener todas las películas y mostrar los actores que han actuado en ellas, incluso si algunas películas no tienen actores asociados.

```
SELECT
  f.film_id,
  f.title,
  a.actor_id,
  a.first_name,
  A.last_name
FROM
  film f
LEFT JOIN
  film_actor fa ON f.film_id = fa.film_id
LEFT JOIN
  actor a ON fa.actor_id = a.actor_id;
```

32. Obtener todos los actores y mostrar las películas en las que han actuado, incluso si algunos actores no han actuado en ninguna película.

```
SELECT
  a.actor_id,
  a.first_name,
  a.last_name,
  f.film_id,
  f.title
FROM
  actor a
LEFT JOIN
  film_actor fa ON a.actor_id = fa.actor_id
LEFT JOIN
  film f ON fa.film_id = f.film_id;
```

33. Obtener todas las películas que tenemos y todos los registros de alquiler.

```
SELECT
  f.film_id,
  f.title,
  i.inventory_id,
  r.rental_id,
  r.rental_date,
  r.return_date
FROM
  film f
LEFT JOIN
  inventory i ON f.film_id = i.film_id
LEFT JOIN
  rental r ON i.inventory_id = r.inventory_id;
```

34. Encuentra los 5 clientes que más dinero se hayan gastado con nosotros.

```
SELECT c.customer_id, c.first_name, c.last_name, SUM(p.amount) AS
total_gastado
FROM customer c
JOIN payment p ON c.customer_id = p.customer_id
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_gastado DESC
LIMIT 5;
```

35. Selecciona todos los actores cuyo primer nombre es 'Johnny'.

```
SELECT *
FROM actor
WHERE first_name = 'JOHNNY';
```

36. Renombra la columna “first_name” como Nombre y “last_name” como Apellido.

```
SELECT
  first_name AS Nombre,
  last_name AS Apellido
FROM actor;
```

37. Encuentra el ID del actor más bajo y más alto en la tabla actor.

```
SELECT
  MIN(actor_id) AS ID_menor,
  MAX(actor_id) AS ID_mayor
FROM actor;
```

38. Cuenta cuántos actores hay en la tabla “actor”.

```
SELECT COUNT(*) AS total_actores
FROM actor;
```

39. Selecciona todos los actores y ordénalos por apellido en orden ascendente.

```
SELECT *
FROM actor
ORDER BY last_name ASC;
```

40. Selecciona las primeras 5 películas de la tabla “film”.

```
SELECT *
FROM film
LIMIT 5;
```

41. Agrupa los actores por su nombre y cuenta cuántos actores tienen el mismo nombre. ¿Cuál es el nombre más repetido?

```
SELECT first_name, COUNT(*) AS cantidad
FROM actor
GROUP BY first_name
ORDER BY cantidad DESC
LIMIT 1;
```

42. Encuentra todos los alquileres y los nombres de los clientes que los realizaron.

```
SELECT r.rental_id, c.first_name, c.last_name, r.rental_date, r.return_date
FROM rental r
JOIN customer c ON r.customer_id = c.customer_id;
```

43. Muestra todos los clientes y sus alquileres si existen, incluyendo aquellos que no tienen alquileres.

```
SELECT c.customer_id, c.first_name, c.last_name, r.rental_id, r.rental_date, r.return_date
FROM customer c
LEFT JOIN rental r ON c.customer_id = r.customer_id;
```

44. Realiza un CROSS JOIN entre las tablas film y category. ¿Aporta valor esta consulta? ¿Por qué? Deja después de la consulta la contestación.

```
SELECT f.film_id, f.title, c.category_id, c.name
FROM film f
CROSS JOIN category c;
```

45. Encuentra los actores que han participado en películas de la categoría 'Action'.

```
SELECT DISTINCT a.actor_id, a.first_name, a.last_name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Action';
```

46. Encuentra todos los actores que no han participado en películas.

```
SELECT actor_id, first_name, last_name
FROM actor a
WHERE NOT EXISTS (
  SELECT 1
  FROM film_actor fa
  WHERE fa.actor_id = a.actor_id
);
```

47. Selecciona el nombre de los actores y la cantidad de películas en las que han participado.

```
SELECT
  A.first_name,
  A.last_name,
  COUNT(fa.film_id) AS numero_peliculas
FROM
  actor a
LEFT JOIN
  film_actor fa ON a.actor_id = fa.actor_id
GROUP BY
  a.actor_id, a.first_name, a.last_name;
```

48. Crea una vista llamada “actor_num_peliculas” que muestre los nombres de los actores y el número de películas en las que han participado.

```
CREATE VIEW actor_num_peliculas AS

SELECT
  Actor.first_name,
  Actor.last_name,
  COUNT(film_actor.film_id) AS num_peliculas
FROM
  Actor
LEFT JOIN
  film_actor ON actor.actor_id = film_actor.actor_id
GROUP BY
  actor.actor_id, actor.first_name, actor.last_name;
```

49. Calcula el número total de alquileres realizados por cada cliente.

```
SELECT
  c.customer_id,
  c.first_name,
  c.last_name,
  COUNT(r.rental_id) AS total_alquileres
FROM
  customer c
LEFT JOIN
  rental r ON c.customer_id = r.customer_id
GROUP BY
  c.customer_id, c.first_name, c.last_name;
```

50. Calcula la duración total de las películas en la categoría 'Action'.

```sql
SELECT SUM(f.length) AS duracion_total
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Action';
```

51. Crea una tabla temporal llamada “cliente_rentas_temporal” para almacenar el total de alquileres por cliente.

```sql
CREATE TEMPORARY TABLE cliente_rentas_temporal AS
SELECT
  c.customer_id,
  c.first_name,
  c.last_name,
  COUNT(r.rental_id) AS total_alquileres
FROM
  customer c
LEFT JOIN
  rental r ON c.customer_id = r.customer_id
GROUP BY
  c.customer_id, c.first_name, c.last_name;
```

52. Crea una tabla temporal llamada “peliculas_alquiladas” que almacene las películas que han sido alquiladas al menos 10 veces.

```sql
CREATE TEMPORARY TABLE peliculas_alquiladas AS
SELECT
  f.film_id,
  f.title,
  COUNT(r.rental_id) AS cantidad_alquileres
FROM
  film f
JOIN
  inventory i ON f.film_id = i.film_id
JOIN
  rental r ON i.inventory_id = r.inventory_id
GROUP BY
  f.film_id, f.title
HAVING
  COUNT(r.rental_id) >= 10;
```

53. Encuentra el título de las películas que han sido alquiladas por el cliente con el nombre ‘Tammy Sanders’ y que aún no se han devuelto. Ordena los resultados alfabéticamente por título de película.

```
SELECT f.title
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
JOIN customer c ON r.customer_id = c.customer_id
WHERE c.first_name = 'TAMMY' AND c.last_name = 'SANDERS' AND r.return_date
IS NULL
ORDER BY f.title ASC;
```

54. Encuentra los nombres de los actores que han actuado en al menos una película que pertenece a la categoría ‘Sci-Fi’. Ordena los resultados alfabéticamente por apellido.

```
SELECT DISTINCT a.first_name, a.last_name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Sci-Fi'
ORDER BY a.last_name ASC;
```

55. Encuentra el nombre y apellido de los actores que han actuado en películas que se alquilaron después de que la película 'Spartacus Cheaper' se alquilara por primera vez. Ordena los resultados alfabéticamente por apellido.

```sql
SELECT MIN(r.rental_date) AS primera_fecha
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
WHERE f.title = 'SPARTACUS CHEAPER';

SELECT DISTINCT a.first_name, a.last_name
FROM actor a
JOIN film_actor fa ON a.actor_id = fa.actor_id
JOIN film f ON fa.film_id = f.film_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
WHERE r.rental_date > (
  SELECT MIN(r.rental_date)
  FROM rental r
  JOIN inventory i ON r.inventory_id = i.inventory_id
  JOIN film f ON i.film_id = f.film_id
  WHERE f.title = 'SPARTACUS CHEAPER'
)
ORDER BY a.last_name ASC;
```

56. Encuentra el nombre y apellido de los actores que no han actuado en ninguna película de la categoría 'Music'.

```sql
SELECT a.first_name, a.last_name
FROM actor a
WHERE NOT EXISTS (
  SELECT 1
  FROM film_actor fa
  JOIN film f ON fa.film_id = f.film_id
  JOIN film_category fc ON f.film_id = fc.film_id
  JOIN category c ON fc.category_id = c.category_id
  WHERE fa.actor_id = a.actor_id AND c.name = 'Music'
);
```

57. Encuentra el título de todas las películas que fueron alquiladas por más de 8 días.

```sql
SELECT DISTINCT f.title
FROM film f
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
WHERE EXTRACT(day FROM (r.return_date - r.rental_date)) > 8;
```

58. Encuentra el título de todas las películas que son de la misma categoría que 'Animation'.

```sql
SELECT f.title
FROM film f
JOIN film_category fc ON f.film_id = fc.film_id
JOIN category c ON fc.category_id = c.category_id
WHERE c.name = 'Animation';
```

59. Encuentra los nombres de las películas que tienen la misma duración que la película con el título 'Dancing Fever'. Ordena los resultados alfabéticamente por título de película.

```sql
SELECT f2.title
FROM film f1
JOIN film f2 ON f1.length = f2.length
WHERE f1.title = 'DANCING FEVER'
ORDER BY f2.title ASC;
```

60. Encuentra los nombres de los clientes que han alquilado al menos 7 películas distintas. Ordena los resultados alfabéticamente por apellido.

```sql
SELECT c.first_name, c.last_name
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film f ON i.film_id = f.film_id
GROUP BY c.customer_id, c.first_name, c.last_name
HAVING COUNT(DISTINCT f.film_id) >= 7
ORDER BY c.last_name ASC;
```

61. Encuentra la cantidad total de películas alquiladas por categoría y muestra el nombre de la categoría junto con el recuento de alquileres.

```
SELECT c.name AS categoria, COUNT(r.rental_id) AS total_alquileres
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
JOIN inventory i ON f.film_id = i.film_id
JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY c.name;
```

62. Encuentra el número de películas por categoría estrenadas en 2006.

```
SELECT c.name AS categoria, COUNT(*) AS peliculas_2006
FROM category c
JOIN film_category fc ON c.category_id = fc.category_id
JOIN film f ON fc.film_id = f.film_id
WHERE f.release_year = 2006
GROUP BY c.name;
```

63. Obtén todas las combinaciones posibles de trabajadores con las tiendas que tenemos.

```
SELECT s.staff_id, s.first_name, s.last_name, st.store_id
FROM staff s
CROSS JOIN store st;
```

64. Encuentra la cantidad total de películas alquiladas por cada cliente y muestra el ID del cliente, su nombre y apellido junto con la cantidad de películas alquiladas.

```
SELECT c.customer_id, c.first_name, c.last_name, COUNT(DISTINCT r.inventory_id) AS peliculas_alquiladas
FROM customer c
JOIN rental r ON c.customer_id = r.customer_id
JOIN inventory i ON r.inventory_id = i.inventory_id
GROUP BY c.customer_id, c.first_name, c.last_name;
```