Memoria Práctica 2.

Martín Sánchez y César Ramírez. Grupo 1201.

Implementación

1. **dvdreq**

   Este programa permitirá al usuario realizar una serie de consultas sobre la basa de datos dvdrental2.

   -dvdreq customer.

   El usuario podrá consultar toda la información los clientes registrados en la base de datos.
   Esta consulta en SQL sería la siguiente:

```
SELECT customer.customer_id, customer.first_name,
       customer.last_name, customer.create_date,
        address.address, address.address2,
        country.country, city.city, address.district,
       address.postal_code

 FROM customer, address, city, country

WHERE customer.address_id = address.address_id and
        address.city_id = city.city_id and
        city.country_id = country.country_id and
        customer.first_name = ? or
        customer.last_name = ?;
```

   Como se pude observar, esta consulta devuelve toda la información sobre el cliente registrado que coincida con el nombre y/o apellido introducido por el usuario. Las “?” serán luego sustituidas por el nombre y apellido del cliente o los clientes que el usuario busca en la base de datos.
   En esta consulta accedemos a las tablas de customer para obtener el Id, el nombre, el apellido y su fecha de registro, y a las tablas

address, city y country para obtener la dirección completa del cliente en cuestión.

-dvdreq film

El usuario podrá acceder a toda la información sobre las películas registradas en la base de datos que coincidan total o parcialmente con el título introducido por el mismo.
Esta consulta en SQL sería:

```
SELECT film.film_id, film.title, film.release_year,
        film.length, film.description, language.name

 FROM film, language

WHERE language.language_id = film.language_id and
        film.title LIKE ?;
```

En esta consulta accedemos a la tabla film para obtener su Id, título, año de estreno, duración y su descripción, y a la tabla language para obtener el idioma de la película.
La "?" será sustituida por el título de la película introducido por el usuario.
Gracias a LIKE podemos devolver todas las películas que coincidan parcial o totalmente con el título introducido por el usuario.

-dvreq rent

Gracias a esta consulta el usuario podrá obtener la información sobre las películas alquiladas por un cliente en concreto entre dos fechas introducidas.

En SQL la consulta se realizaría de la siguiente manera:

```
SELECT rental.rental_id, rental.rental_date, film.film_id, film.title,
        staff.staff_id, staff.first_name, staff.last_name,
        store.store_id, payment.amount

FROM customer, rental, inventory, film, staff, store, payment

WHERE customer.customer_id = rental.customer_id and
        rental.inventory_id = inventory.inventory_id and
        inventory.film_id = film.film_id and
        rental.staff_id = staff.staff_id and
        staff.store_id = store.store_id and
        rental.rental_date >= '%s' and
        rental.rental_date <= '%s' and
        payment.rental_id = rental.rental_id and
        customer.customer_id = %s

ORDER BY rental.rental_date;
```

En esta consulta el usuario tendrá que introducir 3 cosas. La primera será el Id del cliente del que se quiere sacar la información, y luego un intervalo de tiempo dado por dos fechas.
Esta consulta imprimirá una lista con muchísima información acerca de los alquileres realizados por el cliente entre esas dos fechas. La lista estará ordenada por la fecha en que se realizó el alquiler.

-dvdreq recommend

Esta consulta imprime una lista con las 3 películas más alquiladas, que pertenezcan a la categoría más alquilada por el cliente, y que este todavía no las haya alquilado.

Esta consulta es bastante compleja, y en SQL se realizaría así:

```
CREATE VIEW CountCategory AS
SELECT COUNT(*) AS C, category.category_id

FROM film, film_category, category, customer, rental, inventory

WHERE customer.customer_id = rental.customer_id and
      inventory.inventory_id = rental.inventory_id and
      inventory.film_id = film.film_id and
      film_category.film_id = film.film_id and
      category.category_id = film_category.category_id and
      customer.customer_id = %s ;

GROUP BY category.category_id
```

En esta VIEW obtenemos el número de películas de cada categoría alquiladas por el cliente en cuestión.

```
CREATE VIEW MaxCategory AS
SELECT category_id

FROM CountCategory

WHERE C = (SELECT MAX(C) FROM CountCategory);
```

En esta VIEW seleccionamos la categoría más repetida entre las películas alquiladas por el cliente.

```
CREATE VIEW NotRentedByCust AS
SELECT film.film_id

FROM film, film_category, MaxCategory

WHERE film.film_id = film_category.film_id and
        film_category.category_id = MaxCategory.category_id and
        film.film_id NOT IN (SELECT film.film_id
                             FROM rental, inventory, film, customer
                             WHERE  customer.customer_id=
                                    rental.customer_id and
                                    inventory.inventory_id =
                                    rental.inventory_id and
                                    inventory.film_id = film.film_id
                                    and
                                   customer.customer_id = %s);
```

En esta VIEW localizamos las películas que pertenecen a la categoría más alquilada por el cliente, pero que este aún no ha alquilado.

```
CREATE VIEW MostRentedFilms AS
 SELECT COUNT(*) AS D, NotRentedByCust.film_id

FROM customer, rental, inventory, NotRentedByCust, MaxCategory,
        film_category

WHERE customer.customer_id = rental.customer_id and
        inventory.inventory_id = rental.inventory_id and
        inventory.film_id = NotRentedByCust.film_id and
        MaxCategory.category_id = film_category.category_id and
        NotRentedByCust.film_id = film_category.film_id

GROUP BY NotRentedByCust.film_id;
```

En esta VIEW contamos el número de veces que las películas pertenecientes a la categoría favorita del cliente han sido alquiladas en general, pero que no lo han sido por el cliente introducido.

SELECT film.film_id, film.title

FROM film, MostRentedFilms

WHERE film.film_id = MostRentedFilms.film_id

ORDER BY D DESC LIMIT 3;

DROP VIEW MostRentedFilms;
DROP VIEW NotRentedByCust;
DROP VIEW MaxCategory;
DROP VIEW CountCategory;

Aquí, finalmente, seleccionamos las tres películas más alquiladas que pertenecen a la categoría favorita del cliente introducido, pero que este aún no ha alquilado. Y también eliminaríamos las listas creadas para realizar esta consulta.
En C hicimos una mega consulta que hacía todo lo que estas subconsultas hacen por separado, pero el razonamiento seguido es el mismo que el ahora explicado.

2. **dvdrent**

Este programa permite la gestión de un alquiler.

-dvdrent new

Dará de alta una película, incluye tanto el alquiler como el pago. Esta consulta en SQL sería de la siguiente forma:

SELECT customer.customer_id

FROM customer

WHERE customer.customer_id = %s;

Primeramente chequearíamos si el customer introducido se encuentra en la base de datos.

```
SELECT inventory.film_id, inventory.inventory_id

FROM inventory

WHERE inventory.store_id = ? and
      inventory.film_id = ? and
      inventory.inventory_id <> ALL (SELECT rental.inventory_id
                                     FROM rental
                                     WHERE rental.return_date >
                                           CURRENT_TIMESTAMP);
```

Aquí chequeamos si la película que se quiere alquilar existe en la base de datos y si está disponible para ser alquilada.

```
SELECT staff.staff_id

FROM store, staff

WHERE staff.staff_id = %s and
      store.store_id = %s and
      staff.store_id = store.store_id;
```

Aquí comprobamos que el staff al que se le pretende alquilar trabaje en la misma tienda donde se pretende alquilar la película.

```
SELECT store.store_id

FROM store, customer

WHERE store.store_id = %s and
      customer.customer_id = %s and
      customer.store_id = store.store_id
```

Aquí comprobamos que la tienda en la que está dado de alta el cliente coincida con la tienda donde pretende alquilar la película.

```
INSERT INTO rental (rental_id, rental_date, inventory_id,
                    customer_id, return_date, staff_id, last_update)

 VALUES (DEFAULT, CURRENT_TIMESTAMP, ?, ?,
          CURRENT_TIMESTAMP + INTERVAL '1 month', ?,
           DEFAULT)

 RETURNING rental.rental_id;
```

Una vez chequeado todo lo anterior se registra en la tabla rental un nuevo alquiler de película.
Además hacemos que se devuelva el rental_id que se le acaba de dar al alquiler de la película para poder añadir el pago de dicho alquiler.

```
INSERT INTO payment (payment_id, customer_id, staff_id,
                     rental_id, amount, payment_date)

VALUES (DEFAULT, ?, ?, ?, ?, CURRENT_TIMESTAMP);
```

Por último introducimos a la tabla payment toda la información sobre el pago del alquiler que acabamos de realizar.

-dvdrent remove

Da de baja un alquiler de una película. Esto requerirá eliminar el alquiler y el pago asociado a él.

```
SELECT rental.rental_id

FROM rental ,customer, inventory

WHERE rental.rental_id = %s and
      rental.inventory_id = inventory.inventory_id and
      inventory.film_id = film.film_id;
```

Esta primera consulta si el alquiler que deseamos eliminar se ha hecho alguna vez o si no se ha hecho nunca.

DELETE FROM payment

WHERE payment.rental_id = %s;

Una vez chequeado que el alquiler se ha hecho, eliminamos el pago asociado.

DELETE FROM rental

WHERE rental.rental_id = %s;

Finalmente eliminamos el alquiler de la base de datos.

3. **dvdfilm**

Este programa permitirá borrar una película de la base de datos. La consulta en SQL sería la siguiente:

SELECT film.film_id

FROM film

WHERE film.film_id = %s

En esta consulta chequeamos sin la película está en la base de datos.

DELETE payment

WHERE payment.rental_id IN (SELECT payment.rental_id

FROM payment, rental, inventory

WHERE payment.rental_id =
rental.rental_id and
rental.inventory_id =
inventory.inventory_id and
inventory.film_id = ?);

Aquí eliminamos los pagos de los alquileres hechos a la película que deseamos eliminar de la base de datos.

```
DELETE FROM rental

WHERE rental.rental_id IN (SELECT rental.rental_id

                           FROM rental, inventory

                           WHERE rental.inventory_id =
                                 inventory.inventory_id and
                                 inventory.film_id = ?);
```

Aquí lo que hacemos es eliminar todos los alquileres asociados a la película que queremos eliminar.

```
DELETE FROM inventory

WHERE inventory.inventory_id IN (SELECT inventory.inventory_id

                                 FROM inventory

                                 WHERE inventory.film_id = ?);
```

Eliminamos todos los inventarios asociados a los a la película que queremos eliminar.

```
DELETE FROM film_actor

WHERE film_actor.film_id = ?;


DELETE FROM film_category

WHERE film_category.film_id = ?;


DELETE FROM film

WHERE film.film_id = ?;
```

Finalmente eliminamos la película de todas las tablas restantes. De esta manera no queda rastro de la película en la base de datos.

Evidencias.

1.dvdreq

Aquí comprobamos el correcto funcionamiento de las cuatro primeras consultas.

-dvdreq customer

```
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$ ./dvdreq customer -n James -a Smith
Id Cliente |  Nombre  | Apellido | Fecha de Registro | País | Ciudad | Distrito | Código Postal | Dirección 1 | Dirección 2

         1 |     Mary |    Smith | 2006-02-14 |       Japan |     Sasebo |    Nagasaki |  35200 |       1913 Hanoi Way |
       299 |    James |   Gannon | 2006-02-14 |       Japan |  Hiroshima |   Hiroshima |  52137 | 1635 Kuwana Boulevard |
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$
```

Aquí buscamos los clientes cuyo nombre es James y/o su apellido es Smith. Y eso fue lo que obtuvimos.

-dvdreq film

```
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$ ./dvdreq film The
 Id de la Película | Título | Año de estreno | Duración | Idioma | Descripción | Actores

   163 |         Clyde Theory | 2006 | 139 | English             | A Beautiful Yarn of a Astronaut And a Frisbee who must Over
come a Explorer in A Jet Boat |  Sissy Sobieski    Johnny Cage    Greg Chaplin    Meryl Allen
   886 |       Theory Mermaid | 2006 | 184 | English             | A Fateful Yarn of a Composer And a Monkey who must Vanquish
 a Womanizer in The First Manned Space Station |  Cuba Olivier    Fred Costner    Will Wilson    Mena Hopper    Mary Keitel
Julia Fawcett
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$
```

Aquí obtenemos todas las películas que contienen en su título la cadena “The”. Y el resultado es correcto.

-dvdreq rent

```
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$ ./dvdreq rent 1 10-10-2000 11-11-3000
 Id Alquiler | Fecha Alquiler | Película ID | Título | Empleado ID | Nombre | Apellido | Tienda ID | Precio

  1185 | 2005-06-15 00:54:12 |  611 |         Musketeers Wait |   2 |        Jon |   Stephens |   2 |  5.99
  1422 | 2005-06-15 18:02:53 |  228 |        Detective Vision |   2 |        Jon |   Stephens |   2 |  0.99
  1476 | 2005-06-15 21:08:46 |  308 |          Ferris Mother |   1 |       Mike |    Hillyer |   1 |  9.99
  1725 | 2005-06-16 15:18:57 |  159 |            Closer Bang |   1 |       Mike |    Hillyer |   1 |  4.99
  2308 | 2005-06-18 08:41:48 |   44 |           Attacks Hate |   2 |        Jon |   Stephens |   2 |  4.99
  2363 | 2005-06-18 13:33:59 |  766 |          Savannah Town |   1 |       Mike |    Hillyer |   1 |  0.99
  3284 | 2005-06-21 06:24:45 |  997 |             Youth Kick |   1 |       Mike |    Hillyer |   1 |  3.99
  4526 | 2005-07-08 03:17:05 |  316 |           Fire Wolves |   2 |        Jon |   Stephens |   2 |  5.99
  4611 | 2005-07-08 07:33:56 |  764 |         Saturday Lambs |   2 |        Jon |   Stephens |   2 |  5.99
  5244 | 2005-07-09 13:24:07 |  814 |         Snatch Slipper |   2 |        Jon |   Stephens |   2 |  4.99
  5326 | 2005-07-09 16:38:01 |  174 | Confidential Interview |   1 |       Mike |    Hillyer |   1 |  4.99
  6163 | 2005-07-11 10:13:46 |  294 |    Expecations Natural |   2 |        Jon |   Stephens |   2 |  7.99
  7273 | 2005-07-27 11:31:22 |  539 |             Luck Opus |   1 |       Mike |    Hillyer |   1 |  2.99
  7841 | 2005-07-28 09:04:45 |  243 |        Doors President |   2 |        Jon |   Stephens |   2 |  4.99
  8033 | 2005-07-28 16:18:23 |  929 |     Usual Untouchables |   1 |       Mike |    Hillyer |   1 |  4.99
  8074 | 2005-07-28 17:33:39 |  341 |           Frost Head |   1 |       Mike |    Hillyer |   1 |  0.99
  8116 | 2005-07-28 19:20:07 |  982 |          Women Dorado |   1 |       Mike |    Hillyer |   1 |  0.99
  8326 | 2005-07-29 03:58:49 |   22 |       Amistad Midsummer |   2 |        Jon |   Stephens |   2 |  2.99
  9571 | 2005-07-31 02:42:18 |  480 |        Jeepers Wedding |   2 |        Jon |   Stephens |   2 |  2.99
 10437 | 2005-08-01 08:51:04 |    3 |       Adaptation Holes |   1 |       Mike |    Hillyer |   1 |  4.99
 11299 | 2005-08-02 15:36:52 |  709 |             Racer Egg |   2 |        Jon |   Stephens |   2 |  3.99
 11367 | 2005-08-02 18:01:38 |  315 |       Finding Anaconda |   1 |       Mike |    Hillyer |   1 |  0.99
 11824 | 2005-08-17 12:37:54 |  579 |          Minds Truman |   2 |        Jon |   Stephens |   2 |  4.99
 12250 | 2005-08-18 03:57:29 |  204 |      Dalmations Sweden |   1 |       Mike |    Hillyer |   1 |  0.99
 13068 | 2005-08-19 09:55:16 |  663 |        Patient Sister |   2 |        Jon |   Stephens |   2 |  0.99
 13176 | 2005-08-19 13:56:54 |  490 |         Jumanji Blade |   2 |        Jon |   Stephens |   2 |  2.99
 14762 | 2005-08-21 23:33:57 |  924 |   Unforgiven Zoolander |   1 |       Mike |    Hillyer |   1 |  0.99
 14825 | 2005-08-22 01:27:57 |  317 |  Fireball Philadelphia |   2 |        Jon |   Stephens |   2 |  1.99
 15298 | 2005-08-22 19:41:37 |  317 |  Fireball Philadelphia |   1 |       Mike |    Hillyer |   1 |  2.99
 15315 | 2005-08-22 20:03:46 |   70 |       Bikini Borrowers |   2 |        Jon |   Stephens |   2 |  5.99
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$
```

Aquí obtenemos todas las películas alquiladas entre 10/10/2000, y el 11/11/3000, y se observa el correcto funcionamiento.

-dvdreq recommend

```
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$ ./dvdreq recommend 1
 Id de la película | Título | Categoría

   891 |       Timberland Sky | Classics
   358 |       Gilmore Boiled | Classics
   951 |       Voyage Legally | Classics
e400156@2-19-2-19:~/Downloads/EDAT-master/Practica 2$
```

Este es el resultado que obtenemos al ejecutar la recommend query para cliente con Id 1.

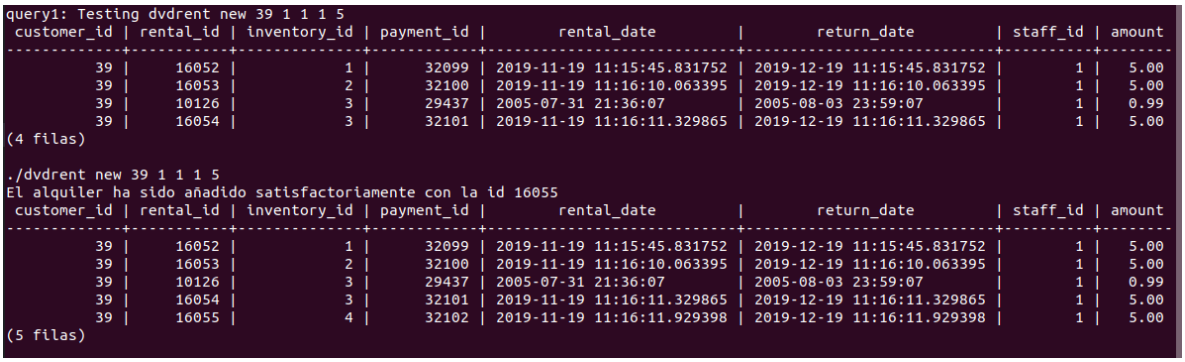
2.dvdrent

-dvdrent new

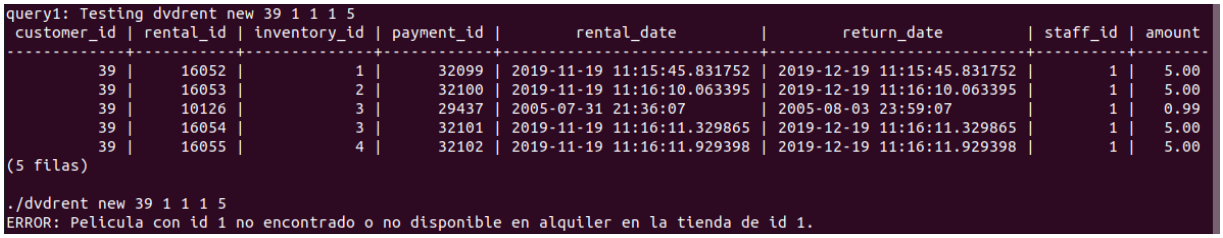
Para comprobar el correcto funcionamiento del programa desarrollamos una query, a la que llamamos query1, que devuelve

los alquileres que ha hecho un cliente, en una tienda para una película determinada, en este caso 39, 1, 1 respectivamente.
La consulta en SQL sería la siguiente:

```
SELECT rental.customer_id, rental.rental_id, inventory.inventory_id,
        payment.payment_id, rental.rental_date, rental.return_date,
        rental.staff_id, payment.amount

FROM inventory, rental, payment

WHERE rental.customer_id = 39 and
    rental.inventory_id = inventory.inventory_id and
    rental.rental_id = payment.rental_id and
    inventory.film_id = 1 and
    inventory.store_id = 1;
```



```
query1: Testing dvdrent new 39 1 1 1 5
 customer_id | rental_id | inventory_id | payment_id |        rental_date         |        return_date         | staff_id | amount
-------------+-----------+--------------+------------+----------------------------+----------------------------+----------+--------
          39 |     16052 |            1 |      32099 | 2019-11-19 11:15:45.831752 | 2019-12-19 11:15:45.831752 |        1 |   5.00
          39 |     16053 |            2 |      32100 | 2019-11-19 11:16:10.063395 | 2019-12-19 11:16:10.063395 |        1 |   5.00
          39 |     10126 |            3 |      29437 | 2005-07-31 21:36:07        | 2005-08-03 23:59:07        |        1 |   0.99
          39 |     16054 |            3 |      32101 | 2019-11-19 11:16:11.329865 | 2019-12-19 11:16:11.329865 |        1 |   5.00
(4 filas)

./dvdrent new 39 1 1 1 5
El alquiler ha sido añadido satisfactoriamente con la id 16055
 customer_id | rental_id | inventory_id | payment_id |        rental_date         |        return_date         | staff_id | amount
-------------+-----------+--------------+------------+----------------------------+----------------------------+----------+--------
          39 |     16052 |            1 |      32099 | 2019-11-19 11:15:45.831752 | 2019-12-19 11:15:45.831752 |        1 |   5.00
          39 |     16053 |            2 |      32100 | 2019-11-19 11:16:10.063395 | 2019-12-19 11:16:10.063395 |        1 |   5.00
          39 |     10126 |            3 |      29437 | 2005-07-31 21:36:07        | 2005-08-03 23:59:07        |        1 |   0.99
          39 |     16054 |            3 |      32101 | 2019-11-19 11:16:11.329865 | 2019-12-19 11:16:11.329865 |        1 |   5.00
          39 |     16055 |            4 |      32102 | 2019-11-19 11:16:11.929398 | 2019-12-19 11:16:11.929398 |        1 |   5.00
(5 filas)
```



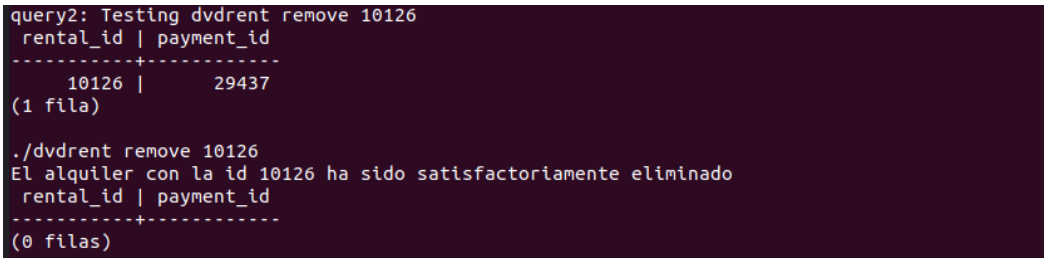
```
query1: Testing dvdrent new 39 1 1 1 5
 customer_id | rental_id | inventory_id | payment_id |        rental_date         |        return_date         | staff_id | amount
-------------+-----------+--------------+------------+----------------------------+----------------------------+----------+--------
          39 |     16052 |            1 |      32099 | 2019-11-19 11:15:45.831752 | 2019-12-19 11:15:45.831752 |        1 |   5.00
          39 |     16053 |            2 |      32100 | 2019-11-19 11:16:10.063395 | 2019-12-19 11:16:10.063395 |        1 |   5.00
          39 |     10126 |            3 |      29437 | 2005-07-31 21:36:07        | 2005-08-03 23:59:07        |        1 |   0.99
          39 |     16054 |            3 |      32101 | 2019-11-19 11:16:11.329865 | 2019-12-19 11:16:11.329865 |        1 |   5.00
          39 |     16055 |            4 |      32102 | 2019-11-19 11:16:11.929398 | 2019-12-19 11:16:11.929398 |        1 |   5.00
(5 filas)

./dvdrent new 39 1 1 1 5
ERROR: Pelicula con id 1 no encontrado o no disponible en alquiler en la tienda de id 1.
```

Realizamos el proceso de inserción varias veces comprobando su correcto funcionamiento. Y como se puede observar llega un momento en el que el inventario se ha consumido en esa tienda para esa película.
Definimos que cada alquiler tuviese un plazo de devolución de un mes.

-dvdrent remove

Para comprobar el correcto funcionamiento del programa desarrollamos una query, a la que llamamos query2, que devuelve el alquiler y el pago asociado a ese alquiler, a partir de un rental_id válido.
En este caso seleccionamos el rental_id 10126.
La consulta en SQL sería la siguiente:

```
SELECT rental.rental_id, payment.payment_id

FROM rental, payment

WHERE rental.rental_id = payment.rental_id and
        rental.rental_id = 10126;
```



Realizamos el proceso de eliminación y volvemos a ejecutar la consulta. Y como se puede observar el alquiler con la id 10126 ha sido eliminado de la base de datos.

3. dvdfilm

Para comprobar el correcto funcionamiento del programa dvdfilm, creamos una query, a la que llamamos query11 que a partir de un film_id (en este caso 111) comprueba que este Id esté en las tablas film, film_actor y film_category devolviendo el id de la película en cada tabla.

Este coincidirá con el introducido (111) si es que esta película está en estas tablas.
Además esta query también devuelve todos los inventory_id asociados a la película,los rental_id asociados a estos inventarios y todos los pagos asociados a esos alquileres.
La query11 en SQL sería la siguiente:

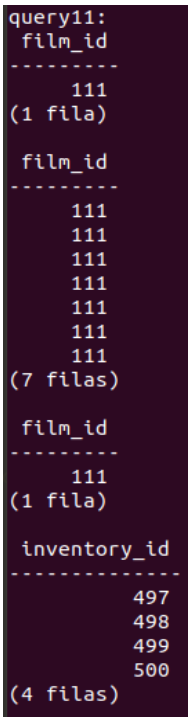
```
SELECT film.film_id

FROM film

WHERE film.film_id = 111;


SELECT film_actor.film_id

FROM film_actor

WHERE film_actor.film_id = 111;


SELECT film_category.film_id

FROM film_category

WHERE film_category.film_id = 111;


SELECT inventory.inventory_id

FROM inventory

WHERE inventory.film_id = 111;


SELECT payment.payment_id

FROM payment, rental, inventory

WHERE payment.rental_id = rental.rental_id and
```

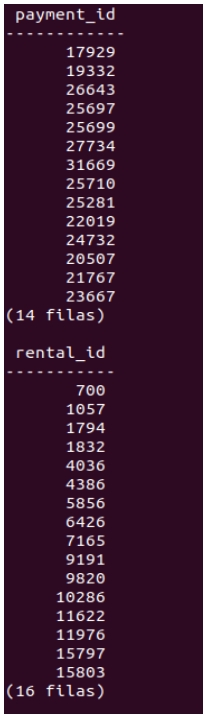
rental.inventory_id = inventory.inventory_id and
inventory.film_id = 111;

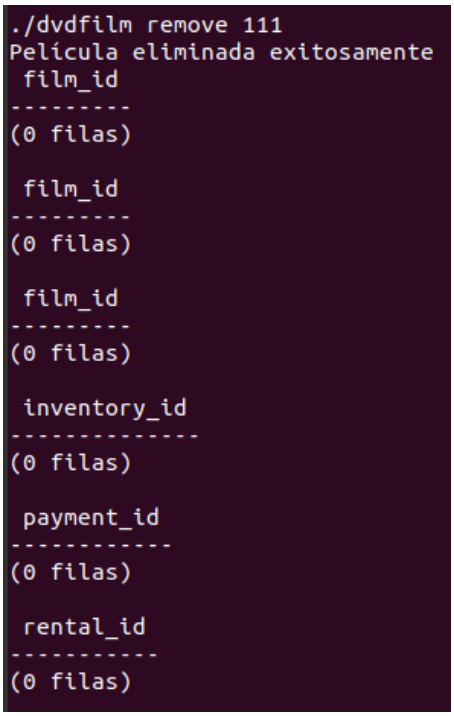
SELECT rental.rental_id

FROM rental, inventory

WHERE rental.inventory_id = inventory.inventory_id and
inventory.film_id = 111;







En estas capturas, gracias a la query11, observamos en la primera el Id de la película en las tablas film, film_actor, film_category y todos los inventory_id asociados a la película que se desea eliminar.
En la segunda observamos todos los rental_id asociados al película en cuestión y todos los pagos asociados a estos alquileres.
En la última captura se observa el resultado tras hacer uso de dvdfilm remove, y se observa como no queda rastro de la película con Id 111, la que queríamos eliminar.