DATA STRUCTURES ASSIGNMENT 2:

DATABASES AND PROGRAMMING LANGUAGES

Samai García González and S. Xiao Fernández Marín
Group 1292
Pair 1

INTRODUCTION

The objective of this assignment is to create our own programs in order to interact with our database from the previous task. For this, we will create several programs in C language that will read our queries in SQL. We will do everything using the ODBC library.

Program 1: *DVDREQ*

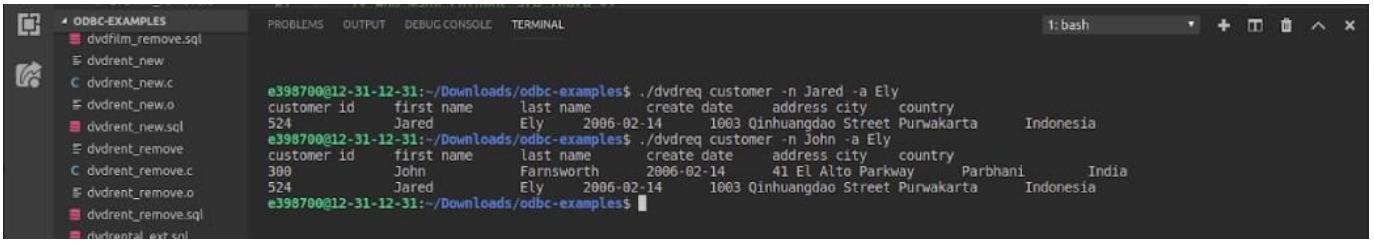
For this program, we wanted it to allow us to perform the following queries on the database:

dvdreq customer -n <First Name> -a <Last Name>

This query first checks the arguments and then prints a list of customer with the same first name, or last name or both, as the ones entered by the user.

```sql
SELECT customer_id,
       first_name,
       last_name,
       create_date,
       address,
       city,
       country
FROM   customer,
       address,
       city,
       country
WHERE  city.country_id = country.country_id
       AND city.city_id = address.city_id
       AND address.address_id = customer.address_id
       AND ( first_name = '?'
             OR last_name = '?' )
```

This is the result when executing it:



dvdreq film <title>

This query, first checks that the title of the film entered as an argument exists:

```
SELECT title
FROM   film
WHERE  title = '%s';
```

Then, if it exists, it prints every movie in the database whose title matches, fully or partially, the entered title:

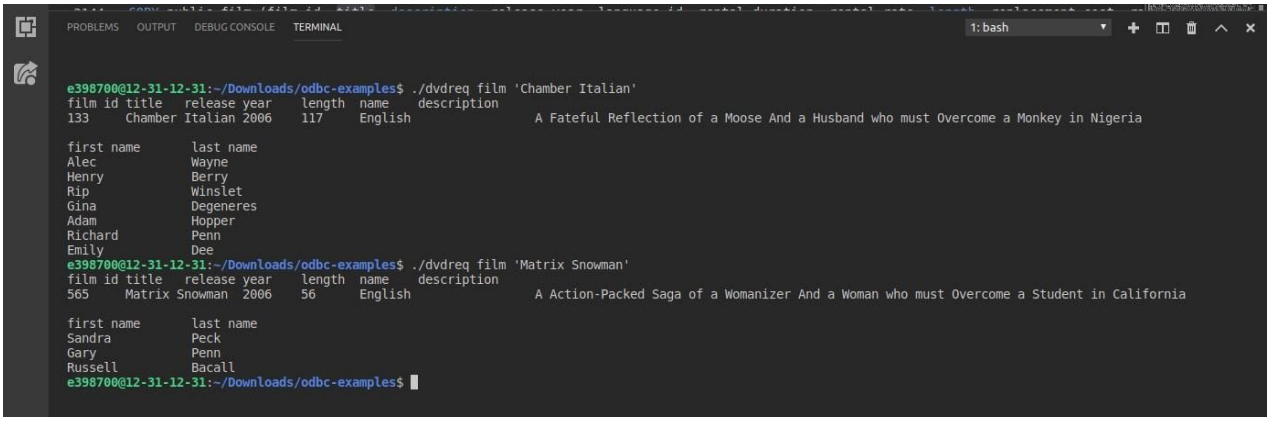
This prints all the information about the film:

```
SELECT film.film_id,
       title,
       release_year,
       length,
       NAME,
       description
FROM   film,
       language,
       film_actor,
       actor
WHERE  language.language_id = film.language_id
       AND actor.actor_id = film_actor.actor_id
       AND film.film_id = film_actor.film_id
       AND film_actor.film_id = film.film_id
       AND title = '?'
```

This prints the first and last name of all the actors of the film.

```
SELECT first_name,
       last_name
FROM   film,
       LANGUAGE,
       film_actor,
       actor
WHERE  LANGUAGE.language_id = film.language_id
       AND actor.actor_id = film_actor.actor_id
       AND film.film_id = film_actor.film_id
       AND film_actor.film_id = film.film_id
       AND title = '%s';
```

This is the result while executing it:



dvdreq rent <customer_id> <init date> <end date>

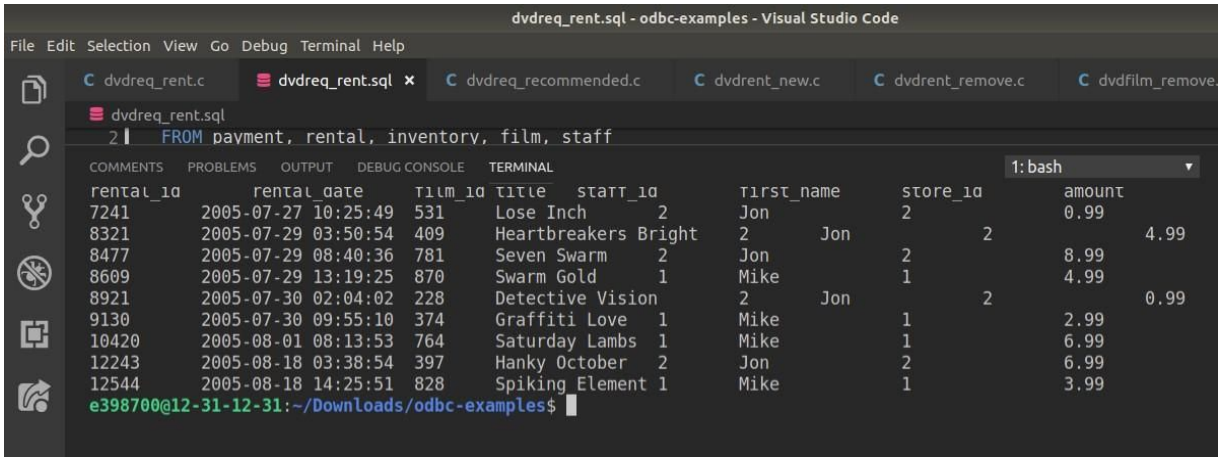
This query first checks that the customer_id received as an argument exists:

```
SELECT customer_id
FROM   rental
WHERE  customer_id = %d;
```

Then, prints the rentals that the customer entered has carried out between the init and end dates entered. It prints the rentals sorted by rental date.

```
SELECT rental.rental_id,
       rental_date,
       film.film_id,
       title,
       staff.staff_id,
       first_name,
       staff.store_id,
       amount
FROM   rental,
       inventory,
       film,
       staff,
       payment
WHERE  rental.inventory_id = inventory.inventory_id
       AND payment.staff_id = staff.staff_id
       AND payment.rental_id = rental.rental_id
       AND inventory.film_id = film.film_id
       AND rental.customer_id = %i
       AND rental_date < '%s'
       AND rental_date > '%s';
```

This is the running result:



dvdreq recommend <customer Id>

First of all, we again check that the customer id exists:

```
SELECT customer_id
FROM   rental
WHERE  customer_id = %d;
```

Then, we create a list with all the film categories that the customer with the entered customer_id has rented:

```
CREATE OR replace VIEW list_rented
AS
  SELECT category_id,
         Count(category_id)
  FROM   film_category,
         (SELECT film_id
          FROM   rental,
                 inventory
          WHERE  rental.customer_id = %i
                 AND rental.inventory_id = inventory.inventory_id) AS
         customer_rented
  WHERE  customer_rented.film_id = film_category.film_id
  GROUP  BY category_id
  ORDER  BY count DESC;
```

After that, it creates a list with the three recommended films that belong to the most rented category of the customer, but that have not been rented before by the customer.

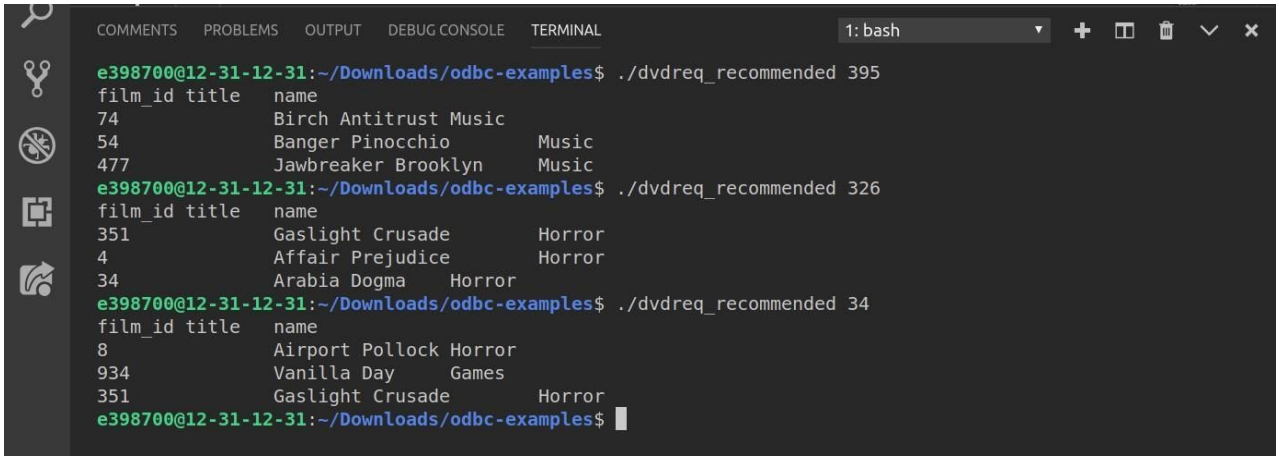
```
CREATE OR replace VIEW most_rented_films AS SELECT film_id
FROM   (
                SELECT   film_category.film_id,
                         Count(rental.inventory_id)
                FROM     film_category,
                         inventory,
                         rental,
                         (
                                SELECT list_rented.category_id
                                FROM   list_rented,
                                       (
                                              SELECT count
                                              FROM   list_rented LIMIT 1) AS max
                                WHERE  list_rented.count = max.count) AS
most_rented_category --This query selects the most rented category from the customer

                WHERE    film_category.category_id = most_rented_category.category_id
                AND      film_category.film_id = inventory.film_id
                AND      inventory.inventory_id = rental.rental_id
                GROUP BY film_category.film_id
                ORDER BY count DESC ) AS x
EXCEPT
```

```
        (
                SELECT film_i)
from    rental,
        inventory
WHERE   rental.customer_id = %i
AND     rental.inventory_id = inventory.inventory_id) limit 3;
```

Finally, it prints the film_id, title and category of the three recommended films.

```
SELECT most_rented_films.film_id,
       film.title,
       category.name
FROM   most_rented_films,
       film,
       film_category,
       category
WHERE  most_rented_films.film_id = film.film_id
       AND film_category.film_id = most_rented_films.film_id
       AND film_category.category_id = category.category_id
```

This is the result of running the program:



Program 2: *DVDRENT*

The objective of this program is to manage the rentals in the database. The following queries were performed:

dvdrent new <customer Id> <film id> <staff id> <store id> <amount>

First checks that there is a film with that film_id, and that the film exists in the inventory so t is available for renting.

```
SELECT film.film_id,
       inventory_id
FROM   film,
       inventory
WHERE  film.film_id = %i
        OR inventory.film_id = %i
```

Then, it checks that the customer_id, staff_id and store_id exist:

```
SELECT customer_id
FROM   customer
WHERE  customer_id = ?;

SELECT staff_id
FROM   staff
WHERE  staff_id = ?;

SELECT store_id
FROM   store
WHERE  store_id = ?;
```

After that, it inserts the values of the arguments in the rental and payment table:

```
INSERT INTO rental
VALUES       (DEFAULT,
              Now(),
              %i,
              %i,
              NULL,
              %i,
              Now())
```

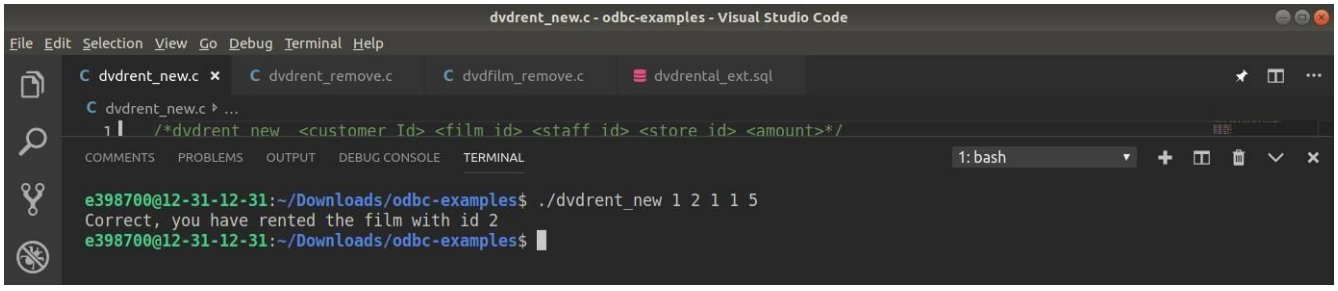
We pick the rental_id from the new rental we created:

```
SELECT rental_id
FROM   rental
WHERE  inventory_id = %s
```

We add it to the payment table:

```
INSERT INTO payment
VALUES      (DEFAULT,
             %i,
             %i,
             %s,
             %i,
             Now())
```

This is the result:



dvdrent remove <rent Id>

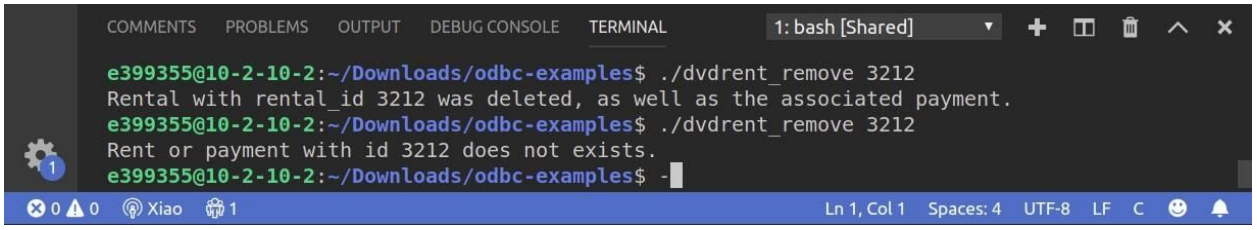
First, we check that the rental_id entered exists in rental or payment:

```
SELECT payment.rental_id FROM rental,
       payment
WHERE  rental.rental_id = payment.rental_id
       AND rental.rental_id = %i
```

Then, it removes the rental associated to that rental_id from the database and its associated payment.

```
DELETE FROM payment
WHERE  payment.rental_id = %i

DELETE FROM rental
WHERE  rental_id = ?
```

This is the result obtained when running it:



Program 3: *DVDFILM*

This program basically has one query, and it is in charge of removing a film from the database.

dvdfilm remove <film id>

First it checks that there is a film in the database with that film_id:

```
SELECT film_id
FROM   film
WHERE  film_id = %i
```

Then, it removes everything related to the film:

- From payment:

```
DELETE FROM payment
WHERE  payment_id IN (SELECT payment_id
                      FROM   payment,
                             rental,
                             inventory
                      WHERE  inventory.inventory_id = rental.inventory_id
                             AND payment.rental_id = rental.rental_id
                             AND film_id = %i);
```

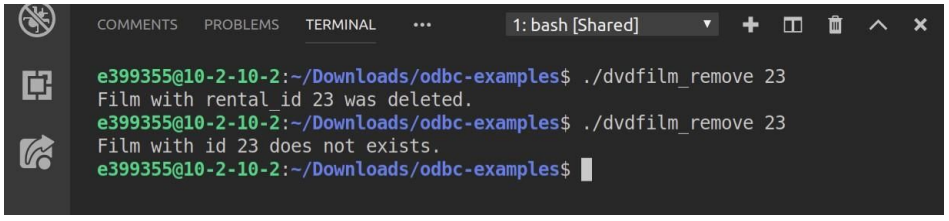
- From rental:

```
DELETE FROM rental
WHERE  rental_id IN (SELECT rental_id
                     FROM   rental,
                            inventory
                     WHERE  inventory.inventory_id = rental.inventory_id
                            AND film_id = %i);
```

- From inventory, film category, film_actor, film:

```
DELETE FROM inventory
WHERE  film_id = %i

DELETE FROM film_category
WHERE  film_id = %i

DELETE FROM film_actor
WHERE  film_id = %i

DELETE FROM film
WHERE  film_id = %i
```

This is the result when running it:



CONCLUSION

In this practice we have learned how to implement queries in sql to c programming. The results obtained where satisfying.