

PROYECTO FINAL BASES DE DATOS

Raúl González Díaz y Miguel Hurtado Mesa

Friends for the Protection of Animals of Granada



ÍNDICE

1. Enunciado

2. Diagrama E/R

3. Paso a tabla

4. Normalización

5. Inserción en Base de Datos

5.1 Tablas

5.2 Inserts

6. Vistas

7. Usuarios

8. Consultas

9. Cursores

10. Triggers

11. Funciones

12. Procedimientos



1. Enunciado

Friends for the Protection of Animals of Granada has hired us to create a database that collect all the data of every section in their web site. The web site has the following sections: shop online, adoptions, donations, members, volunteers and canine residence. Users can register themselves indicating their NIE, name, surname, address, phone number, email and password, or if they are already registered, log in.

Regular users can make occasional donations or become members in the members' section, where they can indicate the amount of money they want to pay, and the way they want to do it, monthly or yearly.

In the volunteers section, the already registered users will find the possibility to offer themselves as homes for abandoned dogs and cats till they find adoption. For this, the future volunteer will have to indicate the kind of housing they have and if they already own other pets. There is another type of volunteers that go to the facilities to help cleaning and feeding animals.

In the shop online section, you can find all the products that the society has. Each product has a value assigned depending on the availability: it will take available if it is ready for delivery, temporarily not available if it is not in stock but the society is waiting for new units to come in a short period of time or not available. Products also have other characteristics like name, code, origin, price, color and size.

Once customers have finished their order, they will get a sheet with all the order's details such as customer information, date of delivery and total cost. At the end, they will have the option to choose between delivery or store-delivery. If they choose the store-delivery option, they will be given an identification number to can track their order whenever the customer wants.

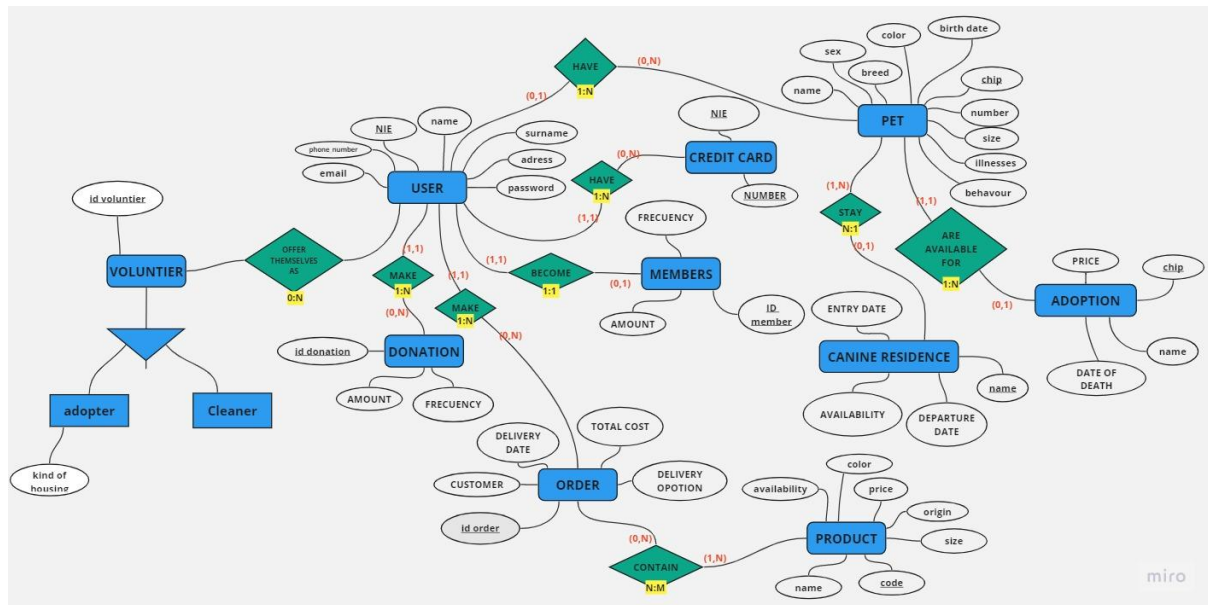
Society will save the credit card number of the customers in its database.

In the canine residence, customers can leave their pets to be supervised when they cannot take care of them. Customers will select the entry and departure dates. If the residence counts with space enough in their facilities, it will be shown as available.

Other case, it will be shown as complete. Customers will have to give some information about the pets: name, sex, chip number, size, breed, race, color, date of birth, presence of illness and behavior and this will create a reservation.

If users are interested in adopting an animal, they can select the animal they are interested in in the adoptions' section. Animals have the same attributes that pets but including a price, which will include all the costs involved in their vaccination and sterilization, and just in case, the date of death.

2. Diagrama E/R



3. Paso a tabla

PET (CHIP, name, sex, birth_date)

USER_have_PET (NIE_USER, CHIP_PET)

USER (NIE, name, surname, phone, email, password, address)

CREDIT_CARD (NIE_USER, NUMBER)

MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)

ADOPTION (CHIP_PET, name, price, date_of_death)

ORDER (ID, NIE_USER, delivery_date, total_cost)

PRODUCT (CODE, name, price, availability, size, color)

ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)

CANINE_RESIDENCE (NAME, availability)

PET_stay_RESIDENCE (CHIP_PET, name_residence)

DONATION (ID, NIE_USER, amount, frequency)

VOLUNTEER (ID_VOLUNTIER, NIE_USER)

ADOPTER (ID_VOLUNTEER, kind_housing)

CLEANER (ID_VOLUNTEER)

4. Normalización

Para normalizar la base de datos, podemos seguir los principios de normalización y dividir las tablas en entidades lógicas más pequeñas para eliminar redundancias y garantizar la integridad de los datos. A continuación, se presenta una posible normalización de la base de datos en varias tablas:

PET (CHIP, name, sex, birth_date)

USER_have_PET (NIE_USER, CHIP_PET)

USER (NIE, name, surname, phone, email, password, address)

CREDIT_CARD (NIE_USER, NUMBER)

MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)

ADOPTION (CHIP_PET, name, price, date_of_death)

ORDER (ID, NIE_USER, delivery_date, total_cost)

PRODUCT (CODE, name, price, availability, size, color)

ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)

CANINE_RESIDENCE (NAME, availability)

PET_stay_RESIDENCE (CHIP_PET, name_residence)

DONATION (ID, NIE_USER, amount, frequency)

VOLUNTEER (ID_VOLUNTEER, NIE_USER)

ADOPTER (ID_VOLUNTEER, kind_housing)

CLEANER (ID_VOLUNTEER)

En esta normalización, se han eliminado las dependencias funcionales entre las tablas originales, dividiéndolas en entidades más pequeñas y relacionándolas mediante claves primarias (PK) y claves foráneas (FK) para mantener la integridad referencial.

Hemos tenido en cuenta que un usuario podría tener más de un teléfono pero hemos considerado que en nuestro proyecto vamos a limitar el número máximo de teléfonos a uno.

Cabe mencionar que esta es solo una posible normalización de la base de datos, y podría haber otras opciones dependiendo de los requerimientos específicos del sistema y el análisis de las relaciones y dependencias de los datos.

5. Inserción en Base de Datos

Primero hemos procedido a crear la base de datos con el siguiente script:

```
-- Creación de la base de datos
CREATE DATABASE ProtectionAnimals;

-- Uso de la base de datos
USE ProtectionAnimals;
```

5.1 Tablas

Para crear las tablas hemos generado el siguiente script:

Es importante mantener este orden para que las tablas puedan crearse correctamente, ya que con las claves foráneas algunas tablas requieren la existencia de otras tablas.

```
-- Creación de la tabla PET
CREATE TABLE PET (
    CHIP VARCHAR(50) PRIMARY KEY,
    name VARCHAR(50),
```

```
sex VARCHAR(10),
birth_date DATE
);

-- Creación de la tabla USER
CREATE TABLE usuario (
  NIE VARCHAR(50) PRIMARY KEY,
  name VARCHAR(50),
  surname VARCHAR(50),
  phone VARCHAR(20),
  email VARCHAR(100),
  password VARCHAR(100),
  address VARCHAR(100)
);

-- Creación de la tabla USER_have_PET
CREATE TABLE usuario_have_PET (
  NIE_usuario VARCHAR(50),
  CHIP_PET VARCHAR(50),
  FOREIGN KEY (NIE_usuario) REFERENCES usuario(NIE),
  FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP),
  PRIMARY KEY (NIE_usuario, CHIP_PET)
);

-- Creación de la tabla CREDIT_CARD
CREATE TABLE CREDIT_CARD (
  NIE_usuario VARCHAR(50),
  numero VARCHAR(20),
  FOREIGN KEY (NIE_usuario) REFERENCES usuario(NIE),
  PRIMARY KEY (NIE_usuario)
);

-- Creación de la tabla MEMBERS
CREATE TABLE MEMBERS (
  ID_MEMBER INT PRIMARY KEY,
  NIE_USER VARCHAR(50),
  frequency VARCHAR(20),
  amount DECIMAL(10, 2),
```



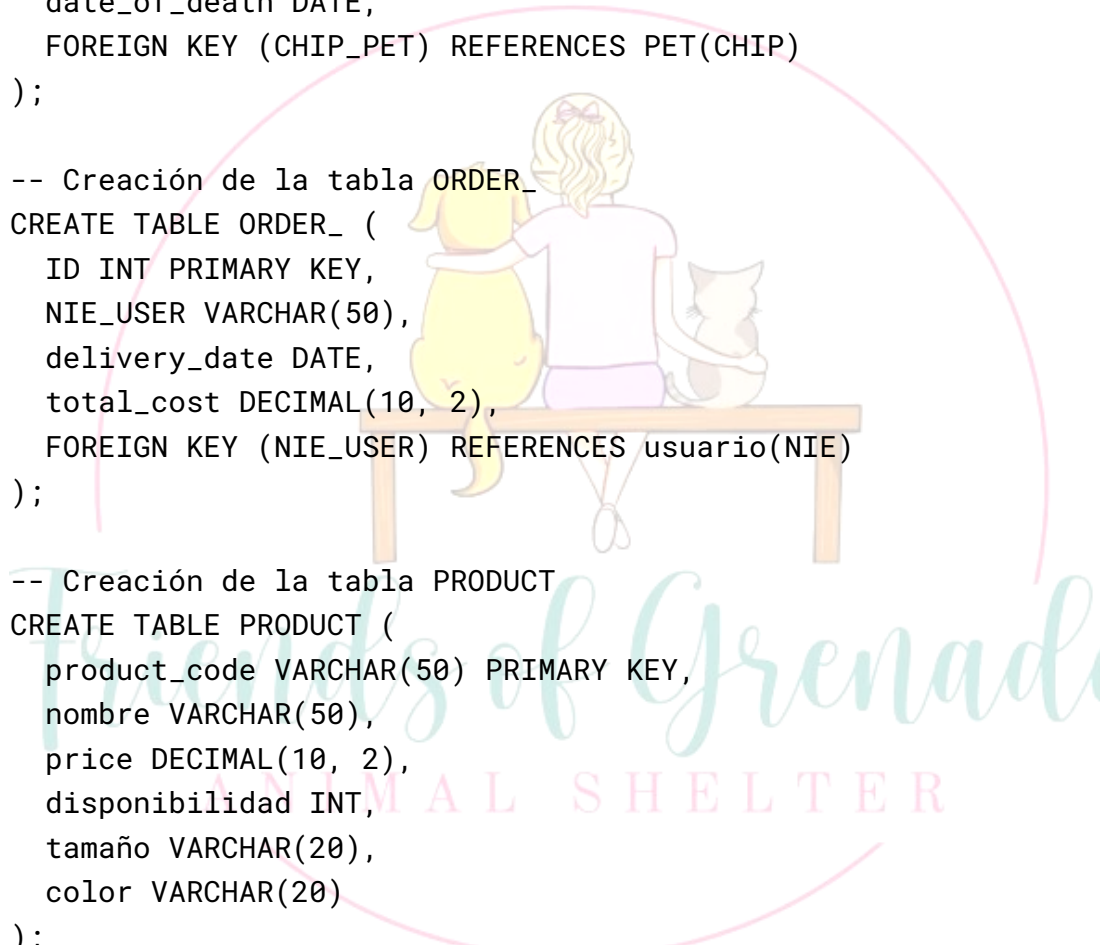
```
FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla ADOPTION
CREATE TABLE ADOPTION (
  CHIP_PET VARCHAR(50) PRIMARY KEY,
  name VARCHAR(50),
  price DECIMAL(10, 2),
  date_of_death DATE,
  FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP)
);

-- Creación de la tabla ORDER_
CREATE TABLE ORDER_ (
  ID INT PRIMARY KEY,
  NIE_USER VARCHAR(50),
  delivery_date DATE,
  total_cost DECIMAL(10, 2),
  FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla PRODUCT
CREATE TABLE PRODUCT (
  product_code VARCHAR(50) PRIMARY KEY,
  nombre VARCHAR(50),
  price DECIMAL(10, 2),
  disponibilidad INT,
  tamaño VARCHAR(20),
  color VARCHAR(20)
);

-- Creación de la tabla ORDER_PRODUCT
CREATE TABLE ORDER_PRODUCT (
  ID_ORDER INT,
  PRODUCT_CODE VARCHAR(50),
  FOREIGN KEY (ID_ORDER) REFERENCES ORDER_(ID),
  FOREIGN KEY (PRODUCT_CODE) REFERENCES PRODUCT(product_code),
  PRIMARY KEY (ID_ORDER, PRODUCT_CODE)
);
```

A large, faint watermark logo is centered in the background. It features a pink circular border. Inside the circle, there is a stylized illustration of a person with blonde hair and a pink bow, seen from behind, sitting on a wooden bench. To the left of the person is a yellow dog, and to the right is a white cat. Below the illustration, the text 'Friends of Grenada' is written in a large, light blue, cursive font. Underneath that, the words 'ANIMAL SHELTER' are written in a smaller, pink, sans-serif font.


```
-- Creación de la tabla CANINE_RESIDENCE
CREATE TABLE CANIN_RESIDENCE (
  name VARCHAR(50) PRIMARY KEY,
  availability INT
);

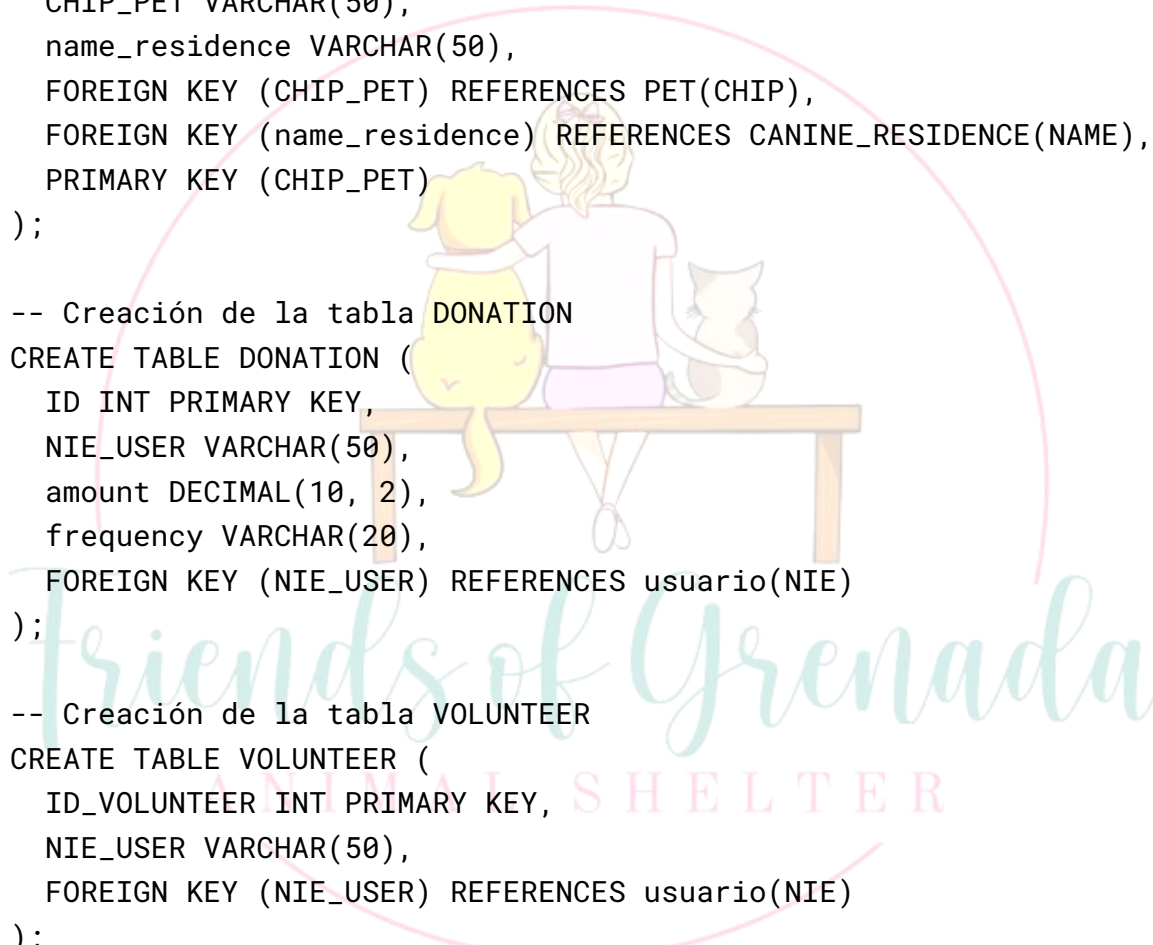
-- Creación de la tabla PET_stay_RESIDENCE
CREATE TABLE PET_stay_RESIDENCE (
  CHIP_PET VARCHAR(50),
  name_residence VARCHAR(50),
  FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP),
  FOREIGN KEY (name_residence) REFERENCES CANINE_RESIDENCE(NAME),
  PRIMARY KEY (CHIP_PET)
);

-- Creación de la tabla DONATION
CREATE TABLE DONATION (
  ID INT PRIMARY KEY,
  NIE_USER VARCHAR(50),
  amount DECIMAL(10, 2),
  frequency VARCHAR(20),
  FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla VOLUNTEER
CREATE TABLE VOLUNTEER (
  ID_VOLUNTEER INT PRIMARY KEY,
  NIE_USER VARCHAR(50),
  FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla ADOPTER
CREATE TABLE ADOPTER (
  ID_VOLUNTEER INT,
  kind_housing VARCHAR(50),
  FOREIGN KEY (ID_VOLUNTEER) REFERENCES VOLUNTEER(ID_VOLUNTEER),
  PRIMARY KEY (ID_VOLUNTEER)
);

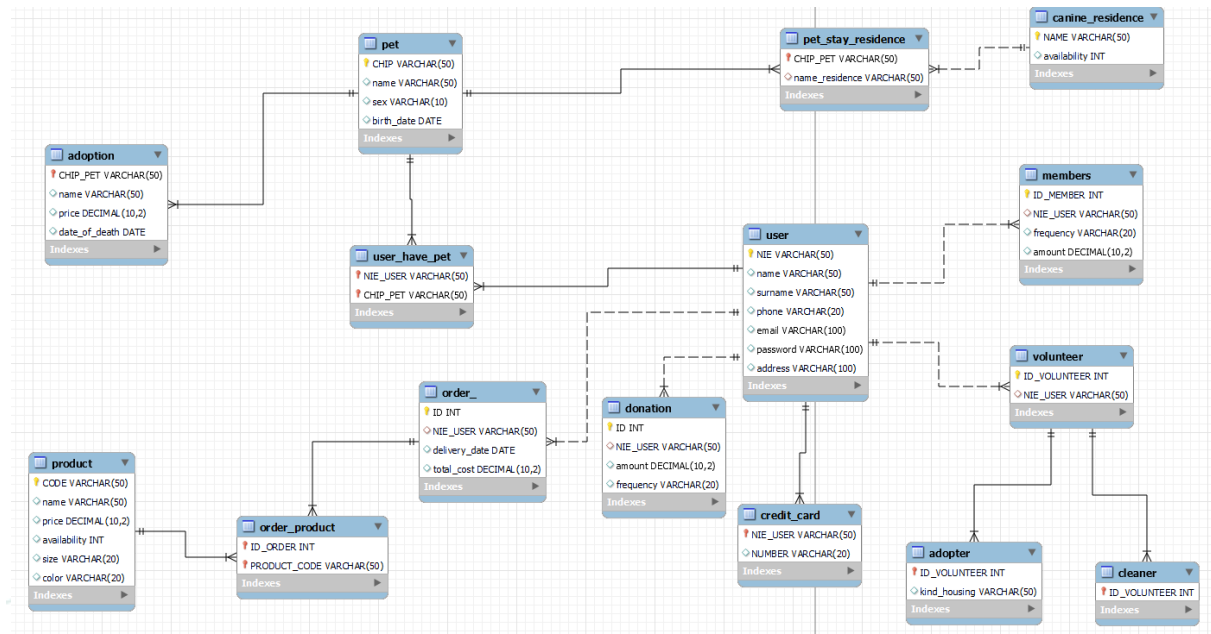
-- Creación de la tabla CLEANER
```



```

CREATE TABLE CLEANER (
    ID_VOLUNTEER INT,
    FOREIGN KEY (ID_VOLUNTEER) REFERENCES VOLUNTEER(ID_VOLUNTEER),
    PRIMARY KEY (ID_VOLUNTEER)
);

```



5.2 Inserts

Para poder trabajar con algunos datos hemos introducido ciertos usuarios, mascotas...etc.

En general se han añadido unas cuantas tuplas por cada tabla para poder realizar posteriormente triggers, consultas, funciones y demás. Así podremos predecir si la base de datos funciona perfectamente.

```

-- Eliminación de la tabla USER_have_PET
DROP TABLE usuario_have_PET;

```

```

-- Eliminación de la tabla CREDIT_CARD
DROP TABLE CREDIT_CARD;

```

```

-- Eliminación de la tabla MEMBERS
DROP TABLE MEMBERS;

```

```
-- Eliminación de la tabla ADOPTION
DROP TABLE ADOPTION;

-- Eliminación de la tabla ORDER_PRODUCT
DROP TABLE ORDER_PRODUCT;

-- Eliminación de la tabla CANINE_RESIDENCE
DROP TABLE CANIN_RESIDENCE;

-- Eliminación de la tabla PET_stay_RESIDENCE
DROP TABLE PET_stay_RESIDENCE;

-- Eliminación de la tabla DONATION
DROP TABLE DONATION;

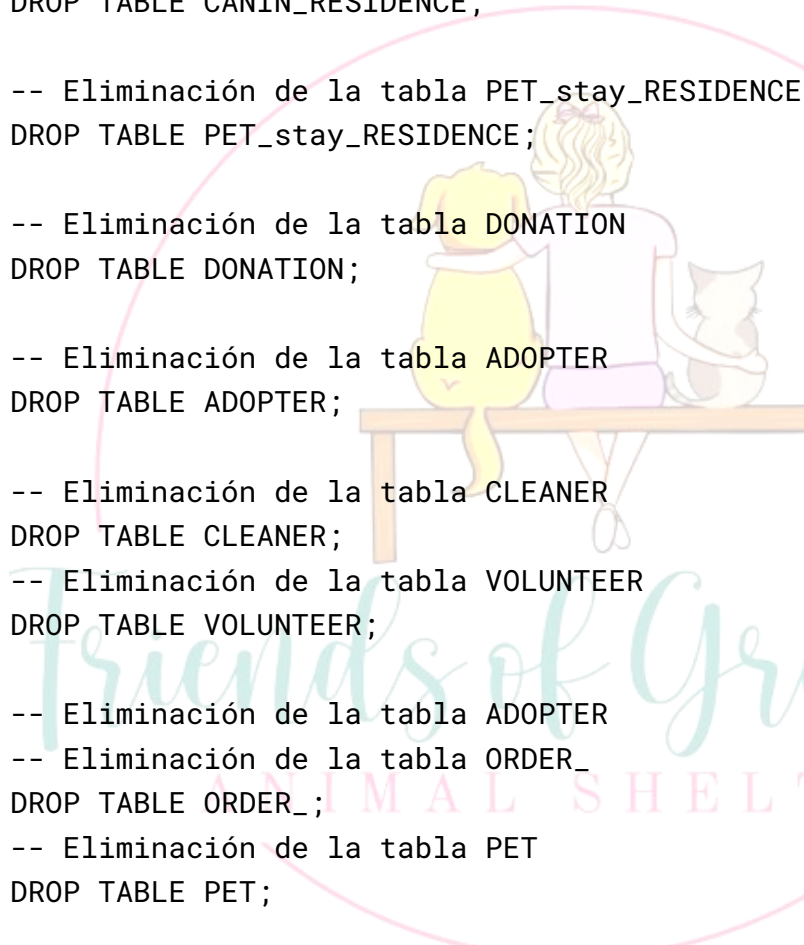
-- Eliminación de la tabla ADOPTER
DROP TABLE ADOPTER;

-- Eliminación de la tabla CLEANER
DROP TABLE CLEANER;
-- Eliminación de la tabla VOLUNTEER
DROP TABLE VOLUNTEER;

-- Eliminación de la tabla ADOPTER
-- Eliminación de la tabla ORDER_
DROP TABLE ORDER_;
-- Eliminación de la tabla PET
DROP TABLE PET;

-- Eliminación de la tabla USER
DROP TABLE usuario;
-- Eliminación de la tabla PRODUCT
DROP TABLE PRODUCT;

-- Insert
-- Inserciones en la tabla PET
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
    ('chip1', 'Max', 'Male', '01-01-2020');
INSERT INTO PET (CHIP, name, sex, birth_date)
```



```
VALUES ('chip2', 'Bella', 'Female', '10-05-2019');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
    ('chip3', 'Charlie', 'Male', '15-08-2018');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
    ('chip4', 'Lucy', 'Female', '28-02-2021');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
    ('chip5', 'Cooper', 'Male', '20-11-2017');

-- Inserciones en la tabla USER
INSERT INTO usuario(NIE, name, surname, phone, email, password,
address)
VALUES
    ('nie1', 'John', 'Doe', '123456789', 'john@example.com',
'password123', '123 Main St');
INSERT INTO usuario(NIE, name, surname, phone, email, password,
address)
VALUES
    ('nie2', 'Jane', 'Smith', '987654321', 'jane@example.com',
'secret456', '456 Elm St');
INSERT INTO usuario(NIE, name, surname, phone, email, password,
address)
VALUES
    ('nie3', 'David', 'Johnson', '555555555', 'david@example.com',
'qwerty789', '789 Oak St');
INSERT INTO usuario(NIE, name, surname, phone, email, password,
address)
VALUES
    ('nie4', 'Emily', 'Davis', '111222333', 'emily@example.com',
'letmein123', '321 Pine St');
INSERT INTO usuario(NIE, name, surname, phone, email, password,
address)
VALUES
    ('nie5', 'Michael', 'Wilson', '999888777', 'michael@example.com',
'password789', '987 Cedar St');

-- Inserciones en la tabla usuario_have_PET
INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
```

```
VALUES
('nie1', 'chip1');
INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES
('nie2', 'chip2');
INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES
('nie3', 'chip3');
INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES
('nie4', 'chip4');
INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES
('nie5', 'chip5');

-- Inserciones en la tabla CREDIT_CARD
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie1', '1234567890123456');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie2', '9876543210987654');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie3', '5555555555555555');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie4', '1111222233334444');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie5', '9999888877776666');

-- Inserciones en la tabla MEMBERS
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(1, 'nie1', 'Monthly', 50.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(2, 'nie2', 'Annual', 200.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
```

```
VALUES
(3, 'nie3', 'Monthly', 40.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(4, 'nie4', 'Annual', 150.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(5, 'nie5', 'Monthly', 30.00);

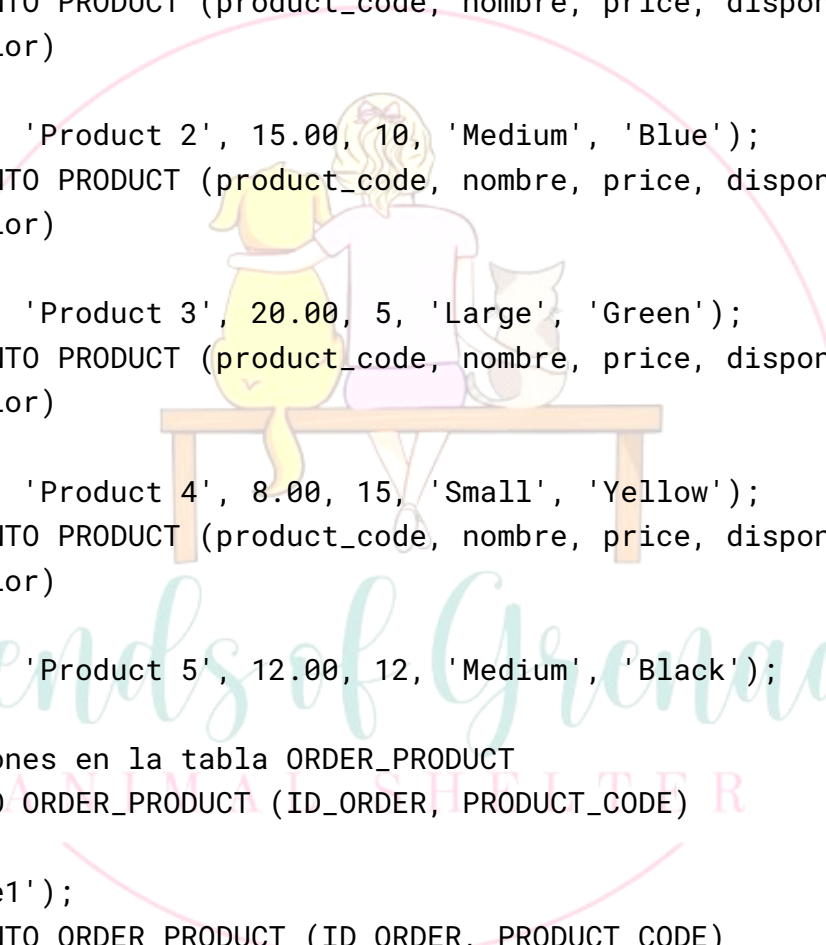
-- Inserciones en la tabla ADOPTION
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip1', 'Max', 100.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip2', 'Bella', 150.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip3', 'Charlie', 75.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip4', 'Lucy', 120.00, '15-04-2022');
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip5', 'Cooper', 90.00, NULL);

-- Inserciones en la tabla ORDER
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
VALUES
(1, 'nie1', '20-05-2023', 75.00);
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
VALUES
(2, 'nie2', '22-05-2023', 150.00);
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
VALUES
(3, 'nie3', '23-05-2023', 100.00);
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
VALUES
(4, 'nie4', '21-05-2023', 50.00);
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
(5, 'nie5', '25-05-2023', 200.00);

-- Inserciones en la tabla PRODUCT
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad,
tamaño, color)
VALUES
('code1', 'Product 1', 10.00, 20, 'Small', 'Red');
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad,
tamaño, color)
VALUES
('code2', 'Product 2', 15.00, 10, 'Medium', 'Blue');
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad,
tamaño, color)
VALUES
('code3', 'Product 3', 20.00, 5, 'Large', 'Green');
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad,
tamaño, color)
VALUES
('code4', 'Product 4', 8.00, 15, 'Small', 'Yellow');
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad,
tamaño, color)
VALUES
('code5', 'Product 5', 12.00, 12, 'Medium', 'Black');

-- Inserciones en la tabla ORDER_PRODUCT
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(1, 'code1');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(1, 'code2');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(2, 'code3');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(3, 'code1');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
```

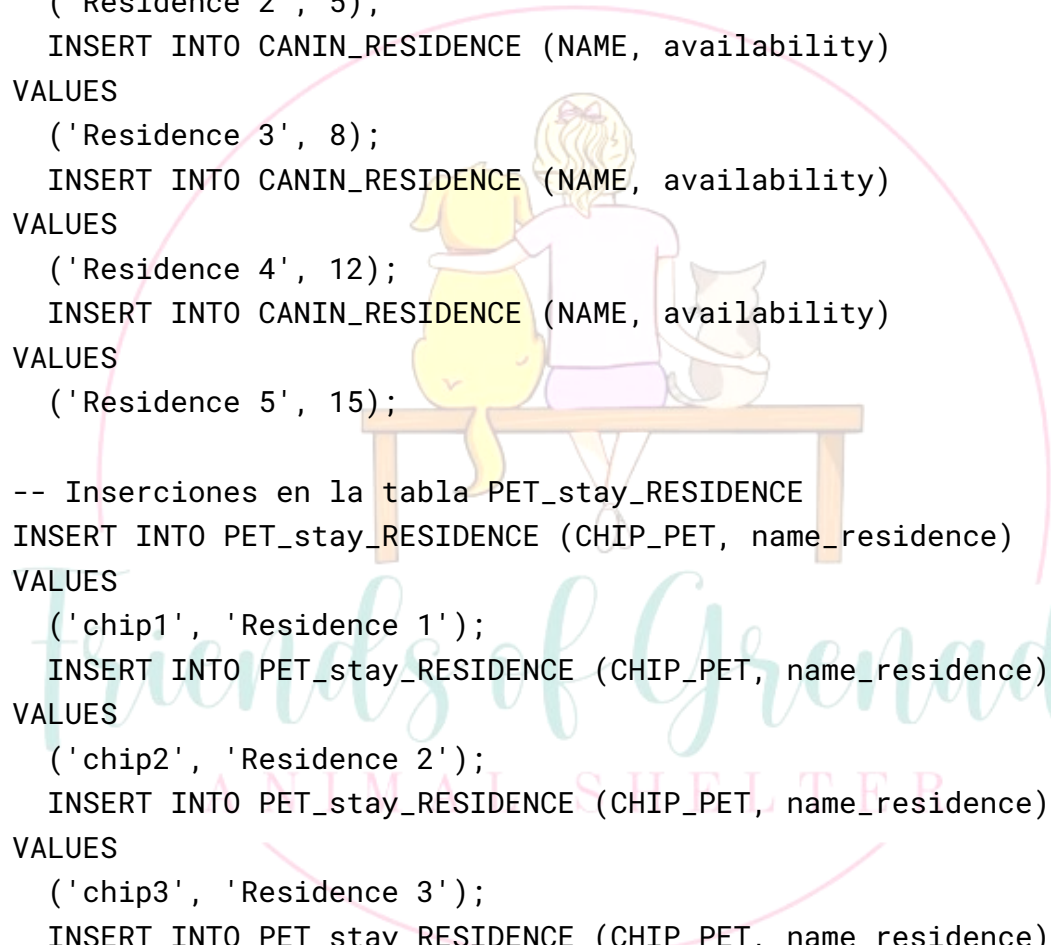



```
(4, 'code4');

-- Inserciones en la tabla CANINE_RESIDENCE
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
    ('Residence 1', 10);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
    ('Residence 2', 5);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
    ('Residence 3', 8);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
    ('Residence 4', 12);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
    ('Residence 5', 15);

-- Inserciones en la tabla PET_stay_RESIDENCE
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
    ('chip1', 'Residence 1');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
    ('chip2', 'Residence 2');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
    ('chip3', 'Residence 3');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
    ('chip4', 'Residence 4');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
    ('chip5', 'Residence 5');

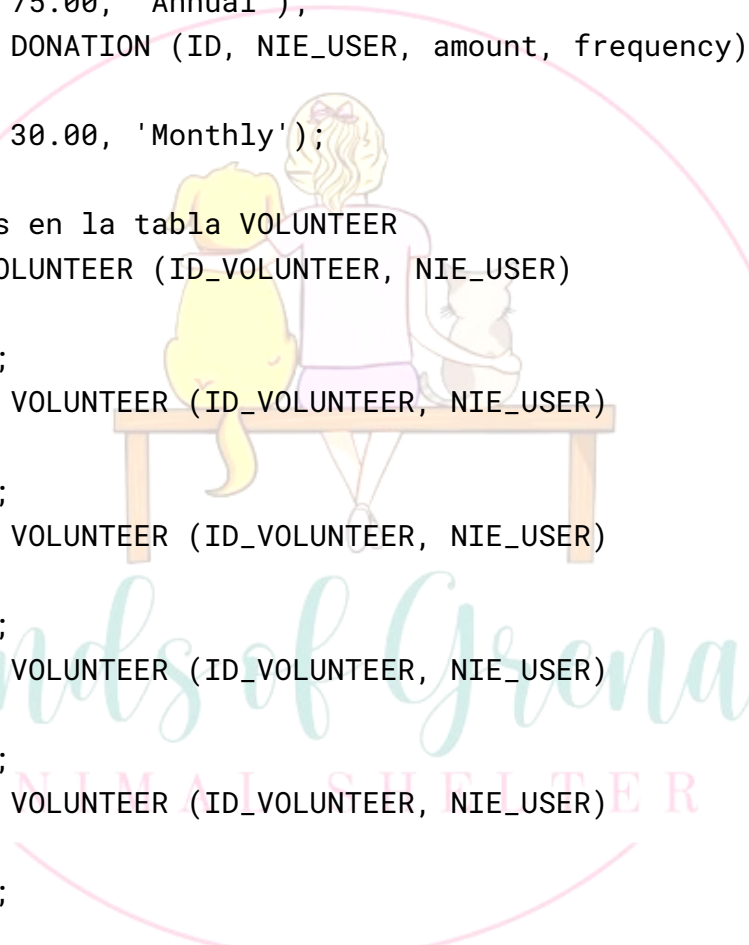
-- Inserciones en la tabla DONATION
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
    (1, 'nie1', 50.00, 'Monthly');
```



```
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(2, 'nie2', 100.00, 'Annual');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(3, 'nie3', 25.00, 'Monthly');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(4, 'nie4', 75.00, 'Annual');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(5, 'nie5', 30.00, 'Monthly');

-- Inserciones en la tabla VOLUNTEER
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(1, 'nie1');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(2, 'nie2');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(3, 'nie3');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(4, 'nie4');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(5, 'nie5');

-- Inserciones en la tabla ADOPTER
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
(1, 'Apartment');
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
(2, 'House');
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
(3, 'Apartment');
```

A large, faint watermark logo is centered on the page. It features a pink circular border. Inside the circle, there is a stylized illustration of a person with long blonde hair, wearing a purple top, sitting on a wooden bench. A yellow dog is sitting to the left of the person, and a white cat is sitting to the right. Below the illustration, the text 'Friends of Granada' is written in a large, cursive, light blue font. Underneath that, the words 'ANIMAL SHELTER' are written in a smaller, pink, sans-serif font.

```
-- Inserciones en la tabla CLEANER
INSERT INTO CLEANER (ID_VOLUNTEER)
VALUES
    (4);
INSERT INTO CLEANER (ID_VOLUNTEER)
VALUES
    (5);
```

6. Vistas

Se han creado dos vistas.

La primera de ellas muestra un conjunto de usuarios con su mascota relacionada:

```
CREATE VIEW Usuario_Mascota AS
SELECT U.name AS user_name, P.name AS pet_name
FROM USER U
INNER JOIN USER_have_PET UP ON U.NIE = UP.NIE_USER
INNER JOIN PET P ON UP.CHIP_PET = P.CHIP;
```

La segunda vista muestra el valor de cada adopción referido a cada uno de los usuarios del sistema:

```
CREATE VIEW PAGO_ADOPCION AS
SELECT u.NIE, u.name, u.surname, SUM(a.price) AS total_paid
FROM usuario u
JOIN usuario_have_PET up ON u.NIE = up.NIE_usuario
JOIN ADOPTION a ON up.CHIP_PET = a.CHIP_PET
GROUP BY u.NIE, u.name, u.surname;
```

7. Usuarios

Existen tres tipos de usuarios en la base de datos:

USUARIO ROOT (Administrador)

Es el usuario principal, tiene completo control de la base de datos, puede editar, modificar, eliminar, compartir e insertar el código que vea necesario. En este caso seríamos nosotros, los desarrolladores del proyecto.

USUARIO CLIENTE

El usuario cliente unicamente puede consultar la tabla productos sin modificar nada.

```
CREATE USER cliente IDENTIFIED BY password DEFAULT TABLESPACE system
QUOTA 100M ON system;
GRANT SELECT ON PRODUCT TO cliente;
```

USUARIO TRABAJADOR

El usuario trabajador puede consultar toda la base de datos pero en ningún momento puede hacer ninguna modificación.

```
CREATE USER trabajador_consulta IDENTIFIED BY password;
GRANT CONNECT, SELECT ANY TABLE TO trabajador_consulta;
```

8. Consultas

A continuación se muestran una serie de consultas para valorar que la base de datos muestra todo en correcto estado:

-- Consultas

-- Mostrar los nombres y fecha de nacimiento de todas las mascotas

```
SELECT name, birth_date
FROM PET;
```

-- Mostrar los productos disponibles

```
SELECT nombre, price
FROM PRODUCT
WHERE disponibilidad > 0;
```

```
-- Mostrar los voluntarios y sus donaciones totales
SELECT v.ID_VOLUNTEER, u.name, COUNT(d.ID)AS donations,
sum(d.amount) AS totalDonado
FROM VOLUNTEER v
JOIN usuario u ON v.NIE_USER = u.NIE
JOIN DONATION d ON v.NIE_USER = d.NIE_USER
GROUP BY v.ID_VOLUNTEER, u.name;

-- Obtener los nombres de las residencias caninas y el número de
mascotas que se están quedando en cada una
SELECT cr.name, COUNT(ps.CHIP_PET) AS pet_count
FROM CANIN_RESIDENCE cr
LEFT JOIN PET_stay_RESIDENCE ps ON cr.name = ps.name_residence
GROUP BY cr.name;

-- Mostrar el nombre y la cantidad total gastada en cada pedido:
SELECT o.ID, u.name, SUM(o.total_cost) AS total_spent
FROM ORDER_ o
JOIN usuario u ON o.NIE_USER = u.NIE
GROUP BY o.ID, u.name;
```

9. Cursores

Hemos introducido dos cursores, el script es el siguiente:

```
-- Cursores
-- Muestra que productos ha pedido cada usuario
SET SERVEROUTPUT ON;
DECLARE
    CURSOR c_pedidos IS
        SELECT u.name AS nombre_usuario, o.NIE_USER, o.ID, p.nombre AS
producto
        FROM ORDER_ o
        JOIN usuario u ON o.NIE_USER = u.NIE
        JOIN ORDER_PRODUCT op ON o.ID = op.ID_ORDER
        JOIN PRODUCT p ON op.PRODUCT_CODE = p.product_code;

v_nombre_usuario usuario.name%TYPE;
```

```

v_usuario ORDER_.NIE_USER%TYPE;
v_pedido_id ORDER_.ID%TYPE;
v_producto PRODUCT.nombre%TYPE;
BEGIN
  OPEN c_pedidos;
  LOOP
    FETCH c_pedidos INTO v_nombre_usuario, v_usuario, v_pedido_id,
v_producto;
    EXIT WHEN c_pedidos%NOTFOUND;
    -- Procesar los datos obtenidos del cursor
    DBMS_OUTPUT.PUT_LINE('Nombre de Usuario: ' || v_nombre_usuario);
    DBMS_OUTPUT.PUT_LINE('Usuario: ' || v_usuario);
    DBMS_OUTPUT.PUT_LINE('Pedido ID: ' || v_pedido_id);
    DBMS_OUTPUT.PUT_LINE('Producto: ' || v_producto);
    DBMS_OUTPUT.PUT_LINE('-----');
  END LOOP;
  CLOSE c_pedidos;
END;

-- información de los usuarios y sus mascotas:
DECLARE
  CURSOR c_usuarios IS
    SELECT u.name, u.surname, p.name AS pet_name
    FROM usuario u
    JOIN usuario_have_PET up ON u.NIE = up.NIE_usuario
    JOIN PET p ON up.CHIP_PET = p.CHIP;

  v_nombre usuario.name%TYPE;
  v_apellido usuario.surname%TYPE;
  v_mascota PET.name%TYPE;
BEGIN
  OPEN c_usuarios;
  LOOP
    FETCH c_usuarios INTO v_nombre, v_apellido, v_mascota;
    EXIT WHEN c_usuarios%NOTFOUND;
    -- Procesar los datos obtenidos del cursor
    DBMS_OUTPUT.PUT_LINE('Usuario: ' || v_nombre || ' ' ||
v_apellido);
    DBMS_OUTPUT.PUT_LINE('Mascota: ' || v_mascota);
  
```

```
        DBMS_OUTPUT.PUT_LINE('-----');
    END LOOP;
    CLOSE c_usuarios;
END;
```

10. Triggers

Para la utilización del primer trigger, hemos creado un trigger que cada vez que se inserta, modifica o se elimina algo en la tabla order, muestra un mensaje por pantalla y actualiza la tabla order.

```
CREATE TABLE OPERATION_LOG (
    OPERATION_ID NUMBER,
    USERNAME VARCHAR2(50),
    OPERATION_DATE DATE
);

CREATE OR REPLACE TRIGGER TRG_LOG_OPERATION
AFTER INSERT OR UPDATE OR DELETE ON ORDER_
FOR EACH ROW
DECLARE
    v_operation_id NUMBER;
BEGIN

    SELECT SEQ_OPERATION_ID.NEXTVAL INTO v_operation_id FROM DUAL;

    INSERT INTO OPERATION_LOG (OPERATION_ID, USERNAME, OPERATION_DATE)
    VALUES (v_operation_id, USER, SYSDATE);

    -- Imprimir mensaje en la salida de DBMS_OUTPUT
    DBMS_OUTPUT.PUT_LINE('Se ha registrado una operación en la tabla:
ORDER_');
    DBMS_OUTPUT.PUT_LINE('Usuario: ' || USER);
    DBMS_OUTPUT.PUT_LINE('Fecha: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY
HH24:MI:SS'));

END;
```


El segundo trigger cuando pides un producto, actualiza la disponibilidad:

```
CREATE OR REPLACE TRIGGER control_disponibilidad
AFTER INSERT ON ORDER_PRODUCT
FOR EACH ROW
BEGIN
    UPDATE PRODUCT
    SET disponibilidad = disponibilidad - 1
    WHERE product_code = :NEW.PRODUCT_CODE;
END;
```

11. Funciones

-- Funcion que calcula el total donado de un usuario

```
CREATE OR REPLACE FUNCTION TOTAL_DONADO(NIE_usuario IN VARCHAR)
RETURN DECIMAL
IS
    total DECIMAL(10, 2);
BEGIN
    SELECT SUM(amount) INTO total
    FROM DONATION
    WHERE NIE_USER = NIE_usuario;

    RETURN total;
END;
```

Para comprobar que la función funcione hemos desarrollado este programa:

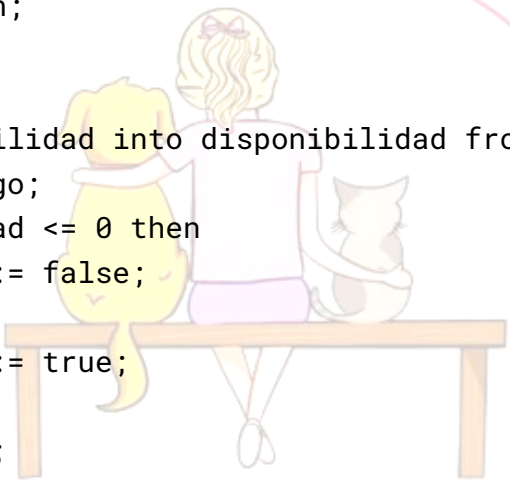
```
DECLARE
    nie_usuario VARCHAR(10) := 'ABC123'; -- Valor de ejemplo para
    NIE_usuario

    total_donado DECIMAL(10, 2);
BEGIN
    total_donado := TOTAL_DONADO(nie_usuario);
```

```
DBMS_OUTPUT.PUT_LINE('Total donado por ' || nie_usuario || ': ' ||  
total_donado);  
END;
```

--Función que verifica si un producto está disponible

```
CREATE OR REPLACE FUNCTION CHECK_PRODUCT_DISPONIBILIDAD(codigo  
product.product_code%type)  
RETURN BOOLEAN  
IS  
    disponibilidad int;  
    disponible boolean;  
BEGIN  
  
    Select disponibilidad into disponibilidad from product where  
product_code = codigo;  
    if disponibilidad <= 0 then  
        disponible := false;  
    else  
        disponible := true;  
    end if;  
    RETURN disponible;  
END;
```

A faint, stylized illustration of a girl with blonde hair and a pink bow, sitting on a wooden bench. A yellow dog is sitting to her left, and a white cat is sitting to her right. The background is a light pink circle.

12. Procedimientos

-- Procedimiento para realizar pedido

```
CREATE OR REPLACE PROCEDURE RealizarPedido(  
    p_order_id IN NUMBER,  
    p_user_nie IN VARCHAR2,  
    p_delivery_date IN DATE,  
    p_product_code IN VARCHAR2,  
    p_quantity IN NUMBER  
)  
AS  
    v_total_cost DECIMAL(10, 2);  
BEGIN
```

```
SELECT price INTO v_total_cost FROM PRODUCT WHERE product_code =  
p_product_code;
```

```
v_total_cost := v_total_cost * p_quantity;
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)  
VALUES (p_order_id, p_user_nie, p_delivery_date, v_total_cost);
```

```
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)  
VALUES (p_order_id, p_product_code);
```

```
DBMS_OUTPUT.PUT_LINE('Pedido realizado con éxito. ID de pedido: '  
|| p_order_id);
```

```
EXCEPTION
```

```
WHEN OTHERS THEN
```

```
DBMS_OUTPUT.PUT_LINE('Error al realizar el pedido: ' ||  
SQLERRM);
```

```
END;
```

```
-- Procedimiento que actualiza el precio de un producto
```

```
CREATE OR REPLACE PROCEDURE ActualizarPrecioProducto(  
  p_product_code IN VARCHAR2,  
  p_new_price IN NUMBER  
)  
AS  
BEGIN
```

```
UPDATE PRODUCT  
SET price = p_new_price  
WHERE product_code = p_product_code;
```

```
DBMS_OUTPUT.PUT_LINE('Precio del producto actualizado con éxito.  
Código de producto: ' || p_product_code);  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
  
        DBMS_OUTPUT.PUT_LINE('No se encontró el producto con el código:  
' || p_product_code);  
  
    WHEN OTHERS THEN  
  
        DBMS_OUTPUT.PUT_LINE('Error al actualizar el precio del  
producto: ' || SQLERRM);  
  
END;
```

13. SCRIPT COMPLETO

- Eliminación de la tabla USER_have_PET
DROP TABLE usuario_have_PET;

-- Eliminación de la tabla CREDIT_CARD
DROP TABLE CREDIT_CARD;

-- Eliminación de la tabla MEMBERS
DROP TABLE MEMBERS;

-- Eliminación de la tabla ADOPTION
DROP TABLE ADOPTION;

-- Eliminación de la tabla ORDER_PRODUCT
DROP TABLE ORDER_PRODUCT;

-- Eliminación de la tabla CANINE_RESIDENCE
DROP TABLE CANIN_RESIDENCE;

-- Eliminación de la tabla PET_stay_RESIDENCE
DROP TABLE PET_stay_RESIDENCE;

-- Eliminación de la tabla DONATION
DROP TABLE DONATION;

DROP TABLE ADOPTER;

-- Eliminación de la tabla CLEANER
DROP TABLE CLEANER;
-- Eliminación de la tabla VOLUNTEER
DROP TABLE VOLUNTEER;

-- Eliminación de la tabla ADOPTER
-- Eliminación de la tabla ORDER_
DROP TABLE ORDER_
-- Eliminación de la tabla PET
DROP TABLE PET;

-- Eliminación de la tabla USER
DROP TABLE usuario;
-- Eliminación de la tabla PRODUCT
DROP TABLE PRODUCT;

-- Creación de la tabla PET
CREATE TABLE PET (
 CHIP VARCHAR(50) PRIMARY KEY,
 name VARCHAR(50),
 sex VARCHAR(10),
 birth_date DATE
);

-- Creación de la tabla USER
CREATE TABLE usuario (
 NIE VARCHAR(50) PRIMARY KEY,
 name VARCHAR(50),
 surname VARCHAR(50),
 phone VARCHAR(20),
 email VARCHAR(100),
 password VARCHAR(100),
 address VARCHAR(100)



```
);  
-- Creación de la tabla USER_have_PET  
CREATE TABLE usuario_have_PET (  
  NIE_usuario VARCHAR(50),  
  CHIP_PET VARCHAR(50),  
  FOREIGN KEY (NIE_usuario) REFERENCES usuario(NIE),  
  FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP),  
  PRIMARY KEY (NIE_usuario, CHIP_PET)  
);
```

```
-- Creación de la tabla CREDIT_CARD  
CREATE TABLE CREDIT_CARD (  
  NIE_usuario VARCHAR(50),  
  numero VARCHAR(20),  
  FOREIGN KEY (NIE_usuario) REFERENCES usuario(NIE),  
  PRIMARY KEY (NIE_usuario)  
);
```

```
-- Creación de la tabla MEMBERS  
CREATE TABLE MEMBERS (  
  ID_MEMBER INT PRIMARY KEY,  
  NIE_USER VARCHAR(50),  
  frequency VARCHAR(20),  
  amount DECIMAL(10, 2),  
  FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)  
);
```

```
-- Creación de la tabla ADOPTION  
CREATE TABLE ADOPTION (  
  CHIP_PET VARCHAR(50) PRIMARY KEY,  
  name VARCHAR(50),  
  price DECIMAL(10, 2),  
  date_of_death DATE,  
  FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP)  
);
```

```
-- Creación de la tabla ORDER_  
CREATE TABLE ORDER_ (  
  ID INT PRIMARY KEY,  
  NIE_USER VARCHAR(50),
```

```
delivery_date DATE,  
total_cost DECIMAL(10, 2),  
FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)  
);
```

-- Creación de la tabla PRODUCT

```
CREATE TABLE PRODUCT (  
product_code VARCHAR(50) PRIMARY KEY,  
nombre VARCHAR(50),  
price DECIMAL(10, 2),  
disponibilidad INT,  
tamaño VARCHAR(20),  
color VARCHAR(20)  
);
```

-- Creación de la tabla ORDER_PRODUCT

```
CREATE TABLE ORDER_PRODUCT (  
ID_ORDER INT,  
PRODUCT_CODE VARCHAR(50),  
FOREIGN KEY (ID_ORDER) REFERENCES ORDER_(ID),  
FOREIGN KEY (PRODUCT_CODE) REFERENCES PRODUCT(product_code),  
PRIMARY KEY (ID_ORDER, PRODUCT_CODE)  
);
```

-- Creación de la tabla CANINE_RESIDENCE

```
CREATE TABLE CANINE_RESIDENCE (  
name VARCHAR(50) PRIMARY KEY,  
availability INT  
);
```

-- Creación de la tabla PET_stay_RESIDENCE

```
CREATE TABLE PET_stay_RESIDENCE (  
CHIP_PET VARCHAR(50),  
name_residence VARCHAR(50),  
FOREIGN KEY (CHIP_PET) REFERENCES PET(CHIP),  
FOREIGN KEY (name_residence) REFERENCES CANINE_RESIDENCE(NAME),  
PRIMARY KEY (CHIP_PET)  
);
```

-- Creación de la tabla DONATION

```
CREATE TABLE DONATION (  
ID INT PRIMARY KEY,  
NIE_USER VARCHAR(50),
```



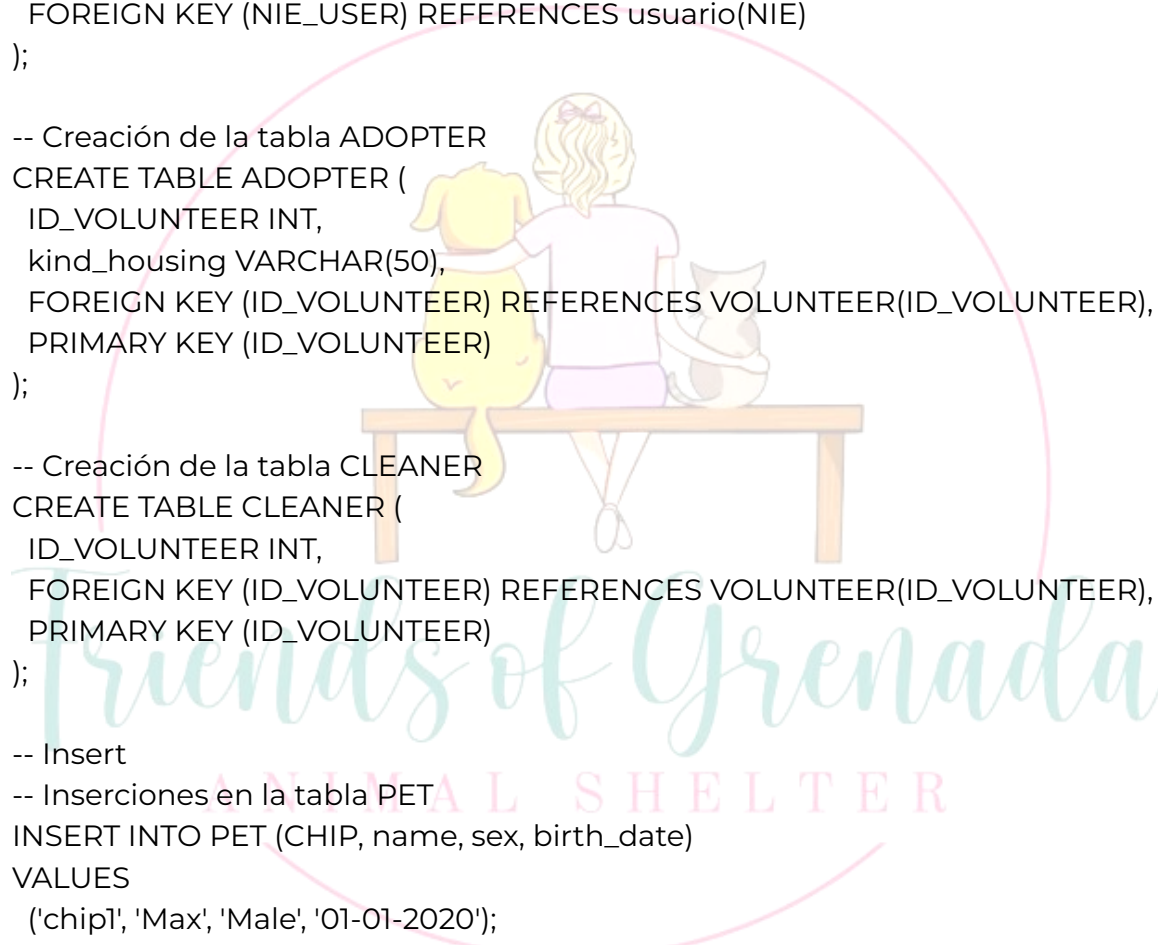
```
amount DECIMAL(10, 2),
frequency VARCHAR(20),
FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla VOLUNTEER
CREATE TABLE VOLUNTEER (
  ID_VOLUNTEER INT PRIMARY KEY,
  NIE_USER VARCHAR(50),
  FOREIGN KEY (NIE_USER) REFERENCES usuario(NIE)
);

-- Creación de la tabla ADOPTER
CREATE TABLE ADOPTER (
  ID_VOLUNTEER INT,
  kind_housing VARCHAR(50),
  FOREIGN KEY (ID_VOLUNTEER) REFERENCES VOLUNTEER(ID_VOLUNTEER),
  PRIMARY KEY (ID_VOLUNTEER)
);

-- Creación de la tabla CLEANER
CREATE TABLE CLEANER (
  ID_VOLUNTEER INT,
  FOREIGN KEY (ID_VOLUNTEER) REFERENCES VOLUNTEER(ID_VOLUNTEER),
  PRIMARY KEY (ID_VOLUNTEER)
);

-- Insert
-- Inserciones en la tabla PET
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
  ('chip1', 'Max', 'Male', '01-01-2020');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES ('chip2', 'Bella', 'Female', '10-05-2019');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
  ('chip3', 'Charlie', 'Male', '15-08-2018');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
  ('chip4', 'Lucy', 'Female', '28-02-2021');
INSERT INTO PET (CHIP, name, sex, birth_date)
VALUES
  ('chip5', 'Cooper', 'Male', '20-11-2017');
```



-- Inserciones en la tabla USER

INSERT INTO usuario(NIE, name, surname, phone, email, password, address)
VALUES

('nie1', 'John', 'Doe', '123456789', 'john@example.com', 'password123', '123 Main St');

INSERT INTO usuario(NIE, name, surname, phone, email, password, address)
VALUES

('nie2', 'Jane', 'Smith', '987654321', 'jane@example.com', 'secret456', '456 Elm St');

INSERT INTO usuario(NIE, name, surname, phone, email, password, address)
VALUES

('nie3', 'David', 'Johnson', '555555555', 'david@example.com', 'qwerty789', '789 Oak St');

INSERT INTO usuario(NIE, name, surname, phone, email, password, address)
VALUES

('nie4', 'Emily', 'Davis', '111222333', 'emily@example.com', 'letmein123', '321 Pine St');

INSERT INTO usuario(NIE, name, surname, phone, email, password, address)
VALUES

('nie5', 'Michael', 'Wilson', '999888777', 'michael@example.com', 'password789', '987 Cedar St');

-- Inserciones en la tabla usuario_have_PET

INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES

('nie1', 'chip1');

INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES

('nie2', 'chip2');

INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES

('nie3', 'chip3');

INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES

('nie4', 'chip4');

INSERT INTO usuario_have_PET (NIE_usuario, CHIP_PET)
VALUES

('nie5', 'chip5');

-- Inserciones en la tabla CREDIT_CARD

INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES

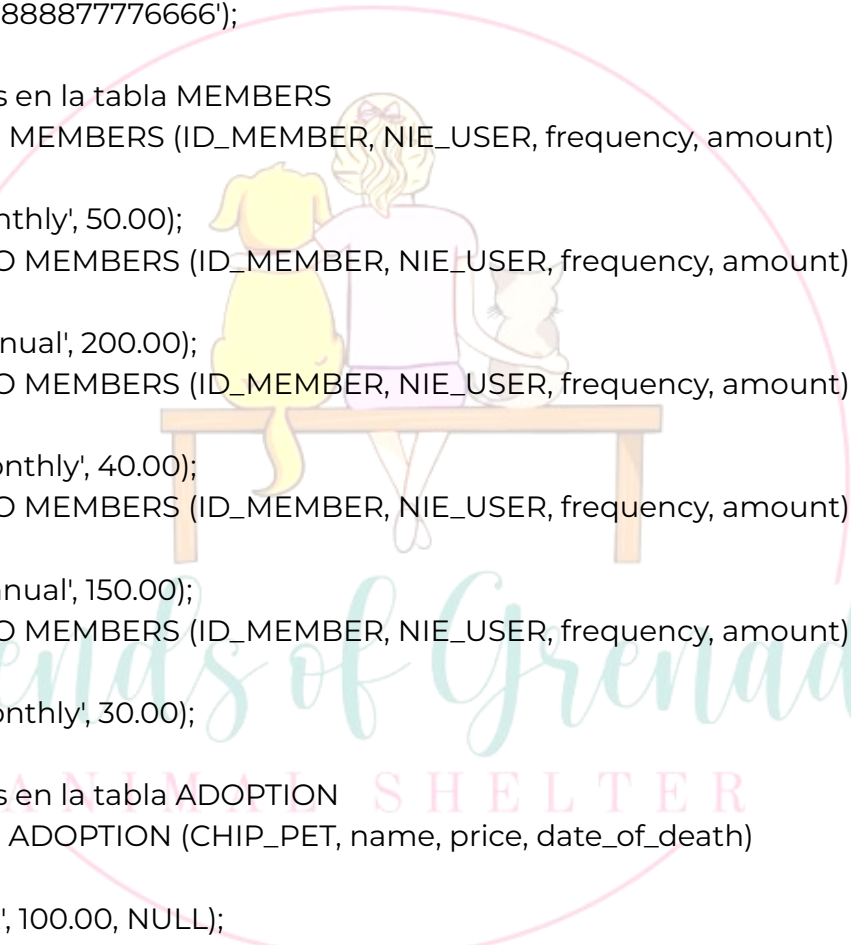
('nie1', '1234567890123456');

INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES

```
('nie2', '9876543210987654');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie3', '5555555555555555');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie4', '1111222233334444');
INSERT INTO CREDIT_CARD (NIE_usuario, numero)
VALUES
('nie5', '9999888877776666');

-- Inserciones en la tabla MEMBERS
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(1, 'nie1', 'Monthly', 50.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(2, 'nie2', 'Annual', 200.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(3, 'nie3', 'Monthly', 40.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(4, 'nie4', 'Annual', 150.00);
INSERT INTO MEMBERS (ID_MEMBER, NIE_USER, frequency, amount)
VALUES
(5, 'nie5', 'Monthly', 30.00);

-- Inserciones en la tabla ADOPTION
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip1', 'Max', 100.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip2', 'Bella', 150.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip3', 'Charlie', 75.00, NULL);
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
('chip4', 'Lucy', 120.00, '15-04-2022');
INSERT INTO ADOPTION (CHIP_PET, name, price, date_of_death)
VALUES
```



```
('chip5', 'Cooper', 90.00, NULL);
```

```
-- Inserciones en la tabla ORDER
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
```

```
(1, 'nie1', '20-05-2023', 75.00);
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
```

```
(2, 'nie2', '22-05-2023', 150.00);
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
```

```
(3, 'nie3', '23-05-2023', 100.00);
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
```

```
(4, 'nie4', '21-05-2023', 50.00);
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
```

```
VALUES
```

```
(5, 'nie5', '25-05-2023', 200.00);
```

```
-- Inserciones en la tabla PRODUCT
```

```
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad, tamaño,  
color)
```

```
VALUES
```

```
('code1', 'Product 1', 10.00, 20, 'Small', 'Red');
```

```
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad, tamaño,  
color)
```

```
VALUES
```

```
('code2', 'Product 2', 15.00, 10, 'Medium', 'Blue');
```

```
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad, tamaño,  
color)
```

```
VALUES
```

```
('code3', 'Product 3', 20.00, 5, 'Large', 'Green');
```

```
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad, tamaño,  
color)
```

```
VALUES
```

```
('code4', 'Product 4', 8.00, 15, 'Small', 'Yellow');
```

```
INSERT INTO PRODUCT (product_code, nombre, price, disponibilidad, tamaño,  
color)
```

```
VALUES
```

```
('code5', 'Product 5', 12.00, 12, 'Medium', 'Black');
```

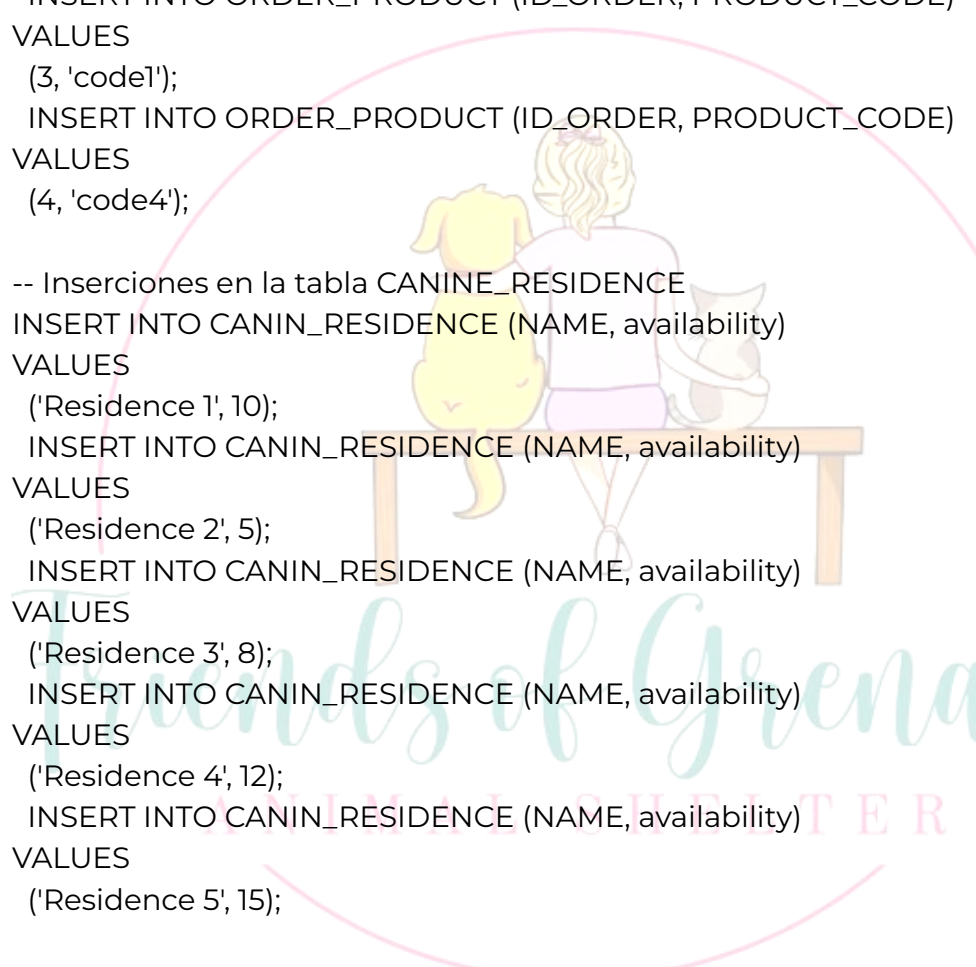
```
-- Inserciones en la tabla ORDER_PRODUCT
```

```
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
```

```
VALUES
(1, 'code1');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(1, 'code2');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(2, 'code3');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(3, 'code1');
INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
VALUES
(4, 'code4');

-- Inserciones en la tabla CANINE_RESIDENCE
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
('Residence 1', 10);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
('Residence 2', 5);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
('Residence 3', 8);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
('Residence 4', 12);
INSERT INTO CANIN_RESIDENCE (NAME, availability)
VALUES
('Residence 5', 15);

-- Inserciones en la tabla PET_stay_RESIDENCE
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
('chip1', 'Residence 1');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
('chip2', 'Residence 2');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
('chip3', 'Residence 3');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
```

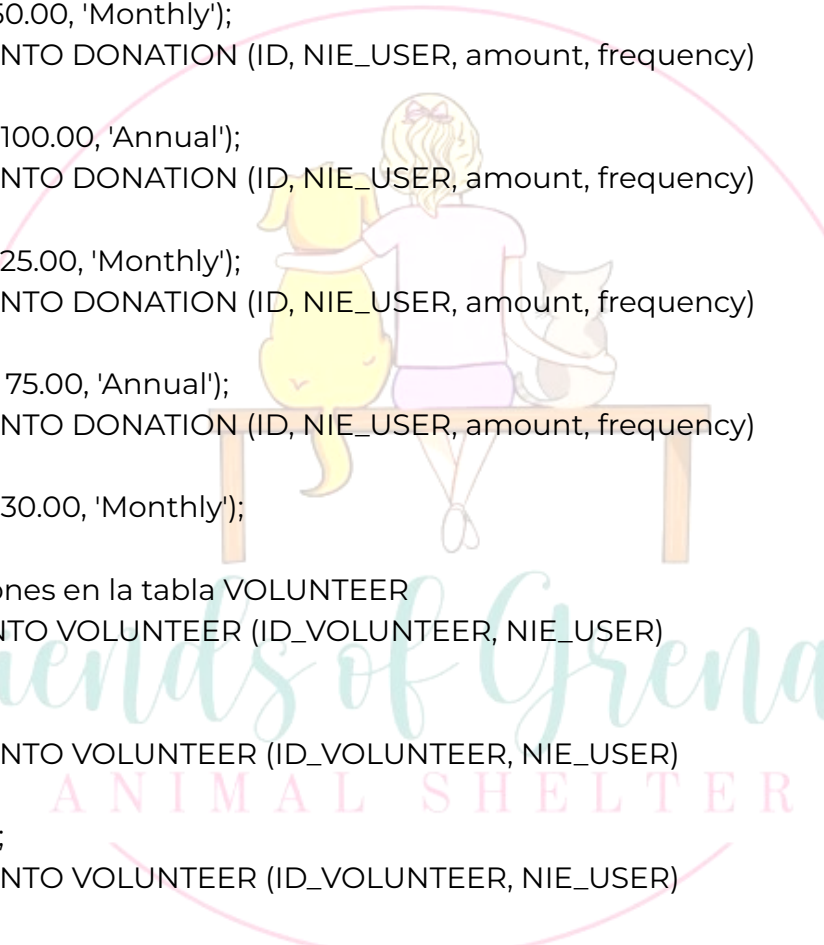


```
VALUES
('chip4', 'Residence 4');
INSERT INTO PET_stay_RESIDENCE (CHIP_PET, name_residence)
VALUES
('chip5', 'Residence 5');

-- Inserciones en la tabla DONATION
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(1, 'nie1', 50.00, 'Monthly');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(2, 'nie2', 100.00, 'Annual');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(3, 'nie3', 25.00, 'Monthly');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(4, 'nie4', 75.00, 'Annual');
INSERT INTO DONATION (ID, NIE_USER, amount, frequency)
VALUES
(5, 'nie5', 30.00, 'Monthly');

-- Inserciones en la tabla VOLUNTEER
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(1, 'nie1');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(2, 'nie2');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(3, 'nie3');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(4, 'nie4');
INSERT INTO VOLUNTEER (ID_VOLUNTEER, NIE_USER)
VALUES
(5, 'nie5');

-- Inserciones en la tabla ADOPTER
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
```



```
(1, 'Apartment');
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
(2, 'House');
INSERT INTO ADOPTER (ID_VOLUNTEER, kind_housing)
VALUES
(3, 'Apartment');

-- Inserciones en la tabla CLEANER
INSERT INTO CLEANER (ID_VOLUNTEER)
VALUES
(4);
INSERT INTO CLEANER (ID_VOLUNTEER)
VALUES
(5);

-- Vistas
drop view usuario_PET;
CREATE VIEW Usuario_PET AS
SELECT usuario.name AS user_name, PET.name AS pet_name
FROM usuario
INNER JOIN usuario_have_PET ON usuario.NIE = usuario_have_PET.NIE_usuario
INNER JOIN PET ON usuario_have_PET.CHIP_PET = PET.CHIP;

select * from usuario_PET;

drop view PAGO_ADOPCION;
CREATE VIEW PAGO_ADOPCION AS
SELECT u.NIE, u.name, u.surname, SUM(a.price) AS total_paid
FROM usuario u
JOIN usuario_have_PET up ON u.NIE = up.NIE_usuario
JOIN ADOPTION a ON up.CHIP_PET = a.CHIP_PET
GROUP BY u.NIE, u.name, u.surname;

select * from PAGO_ADOPCION;

-- usuarios
CREATE USER cliente IDENTIFIED BY password DEFAULT TABLESPACE system
QUOTA 100M ON system;
GRANT SELECT ON PRODUCT TO cliente;

CREATE USER trabajador_consulta IDENTIFIED BY password;
```



```
GRANT CONNECT, SELECT ANY TABLE TO trabajador_consulta;

SELECT username FROM all_users;
-- consultas

-- Mostrar los nombres y fecha de nacimiento de todas las mascotas
SELECT name, birth_date
FROM PET;

-- Mostrar los productos disponibles
SELECT nombre, price
FROM PRODUCT
WHERE disponibilidad > 0;

-- Mostrar los voluntarios y sus donaciones totales
SELECT v.ID_VOLUNTEER, u.name, COUNT(d.ID) AS donations, sum(d.amount) AS
totalDonado
FROM VOLUNTEER v
JOIN usuario u ON v.NIE_USER = u.NIE
JOIN DONATION d ON v.NIE_USER = d.NIE_USER
GROUP BY v.ID_VOLUNTEER, u.name;

-- Obtener los nombres de las residencias caninas y el número de mascotas que
se están quedando en cada una
SELECT cr.name, COUNT(ps.CHIP_PET) AS pet_count
FROM CANIN_RESIDENCE cr
LEFT JOIN PET_stay_RESIDENCE ps ON cr.name = ps.name_residence
GROUP BY cr.name;

-- Mostrar el nombre y la cantidad total gastada en cada pedido:
SELECT o.ID, u.name, SUM(o.total_cost) AS total_spent
FROM ORDER_ o
JOIN usuario u ON o.NIE_USER = u.NIE
GROUP BY o.ID, u.name;

-- cursores
-- muestra que productos ha pedido cada usuario
SET SERVEROUTPUT ON;

DECLARE
CURSOR c_pedidos IS
SELECT u.name AS nombre_usuario, o.NIE_USER, o.ID, p.nombre AS producto
FROM ORDER_ o
```

```

JOIN usuario u ON o.NIE_USER = u.NIE
JOIN ORDER_PRODUCT op ON o.ID = op.ID_ORDER
JOIN PRODUCT p ON op.PRODUCT_CODE = p.product_code;

v_nombre_usuario usuario.name%TYPE;
v_usuario ORDER_.NIE_USER%TYPE;
v_pedido_id ORDER_.ID%TYPE;
v_producto PRODUCT.nombre%TYPE;
BEGIN
OPEN c_pedidos;
LOOP
  FETCH c_pedidos INTO v_nombre_usuario, v_usuario, v_pedido_id, v_producto;
  EXIT WHEN c_pedidos%NOTFOUND;
  -- Procesar los datos obtenidos del cursor
  DBMS_OUTPUT.PUT_LINE('Nombre de Usuario: ' || v_nombre_usuario);
  DBMS_OUTPUT.PUT_LINE('Usuario: ' || v_usuario);
  DBMS_OUTPUT.PUT_LINE('Pedido ID: ' || v_pedido_id);
  DBMS_OUTPUT.PUT_LINE('Producto: ' || v_producto);
  DBMS_OUTPUT.PUT_LINE('-----');
END LOOP;
CLOSE c_pedidos;
END;

-- información de los usuarios y sus mascotas:
DECLARE
CURSOR c_usuarios IS
  SELECT u.name, u.surname, p.name AS pet_name
  FROM usuario u
  JOIN usuario_have_PET up ON u.NIE = up.NIE_usuario
  JOIN PET p ON up.CHIP_PET = p.CHIP;

v_nombre usuario.name%TYPE;
v_apellido usuario.surname%TYPE;
v_mascota PET.name%TYPE;
BEGIN
OPEN c_usuarios;
LOOP
  FETCH c_usuarios INTO v_nombre, v_apellido, v_mascota;
  EXIT WHEN c_usuarios%NOTFOUND;
  -- Procesar los datos obtenidos del cursor
  DBMS_OUTPUT.PUT_LINE('Usuario: ' || v_nombre || ' ' || v_apellido);
  DBMS_OUTPUT.PUT_LINE('Mascota: ' || v_mascota);
  DBMS_OUTPUT.PUT_LINE('-----');

```

```
END LOOP;  
CLOSE c_usuarios;  
END;
```

```
-- triggers
```

```
CREATE TABLE OPERATION_LOG (  
  OPERATION_ID NUMBER,  
  USERNAME VARCHAR2(50),  
  OPERATION_DATE DATE  
);
```

```
CREATE SEQUENCE SEQ_OPERATION_ID;
```

```
CREATE OR REPLACE TRIGGER TRG_LOG_OPERATION  
AFTER INSERT OR UPDATE OR DELETE ON ORDER_  
FOR EACH ROW  
DECLARE  
  v_operation_id NUMBER;  
BEGIN
```

```
  SELECT SEQ_OPERATION_ID.NEXTVAL INTO v_operation_id FROM DUAL;
```

```
  INSERT INTO OPERATION_LOG (OPERATION_ID, USERNAME,  
OPERATION_DATE)  
  VALUES (v_operation_id, USER, SYSDATE);
```

```
  -- Imprimir mensaje en la salida de DBMS_OUTPUT  
  DBMS_OUTPUT.PUT_LINE('Se ha registrado una operación en la tabla: ORDER_');  
  DBMS_OUTPUT.PUT_LINE('Usuario: ' || USER);  
  DBMS_OUTPUT.PUT_LINE('Fecha: ' || TO_CHAR(SYSDATE, 'DD-MON-YYYY  
HH24:MI:SS'));
```

```
END;
```

```
INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)  
VALUES  
(7, 'nie5', '27-05-2023', 300.00);
```

```
SELECT * FROM OPERATION_LOG ;
```

-- cuando pides un producto, actualiza la disponibilidad

```
CREATE OR REPLACE TRIGGER control_disponibilidad
AFTER INSERT ON ORDER_PRODUCT
FOR EACH ROW
BEGIN
    UPDATE PRODUCT
    SET disponibilidad = disponibilidad - 1
    WHERE product_code = :NEW.PRODUCT_CODE;
END;
```

-- funciones

-- funcion que calcula el total donado de un usuario

```
CREATE OR REPLACE FUNCTION TOTAL_DONADO(NIE_usuario VARCHAR)
RETURN DECIMAL
IS
    total DECIMAL(10, 2);
BEGIN
    SELECT SUM(amount) INTO total
    FROM DONATION
    WHERE NIE_USER = NIE_usuario;

    RETURN total;
END;
```

--Función que verifica si un producto está disponible

```
CREATE OR REPLACE FUNCTION CHECK_PRODUCT_DISPONIBILIDAD(codigo
product.product_code%type)
RETURN BOOLEAN
IS
    disponibilidad int;
    disponible boolean;
BEGIN
```

```
    Select disponibilidad into disponibilidad from product where product_code =
codigo;
    if disponibilidad <= 0 then
        disponible := false;
    else
```

```
    disponible := true;
  end if;
  RETURN disponible;
END;

-- procedimientos
-- Procedimiento para realizar pedido
CREATE OR REPLACE PROCEDURE RealizarPedido(
  p_order_id IN NUMBER,
  p_user_nie IN VARCHAR2,
  p_delivery_date IN DATE,
  p_product_code IN VARCHAR2,
  p_quantity IN NUMBER
)
AS
  v_total_cost DECIMAL(10, 2);
BEGIN

  SELECT price INTO v_total_cost FROM PRODUCT WHERE product_code =
  p_product_code;

  v_total_cost := v_total_cost * p_quantity;


  INSERT INTO ORDER_ (ID, NIE_USER, delivery_date, total_cost)
  VALUES (p_order_id, p_user_nie, p_delivery_date, v_total_cost);

  INSERT INTO ORDER_PRODUCT (ID_ORDER, PRODUCT_CODE)
  VALUES (p_order_id, p_product_code);

  DBMS_OUTPUT.PUT_LINE('Pedido realizado con éxito. ID de pedido: ' ||
  p_order_id);
EXCEPTION
  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('Error al realizar el pedido: ' || SQLERRM);

END;
```



```
-- procedimiento que actualiza el precio de un producto
CREATE OR REPLACE PROCEDURE ActualizarPrecioProducto(
  p_product_code IN VARCHAR2,
  p_new_price IN NUMBER
)
AS
BEGIN

  UPDATE PRODUCT
  SET price = p_new_price
  WHERE product_code = p_product_code;

  DBMS_OUTPUT.PUT_LINE('Precio del producto actualizado con éxito. Código de
producto: ' || p_product_code);
EXCEPTION
  WHEN NO_DATA_FOUND THEN

    DBMS_OUTPUT.PUT_LINE('No se encontró el producto con el código: ' ||
p_product_code);

  WHEN OTHERS THEN

    DBMS_OUTPUT.PUT_LINE('Error al actualizar el precio del producto: ' ||
SQLERRM);

END;
```

