M10UF1 - EVIL CORP DATA BASE

Alumno: Marcel Setó
Profesor: Rafa Laguna

ÍNDICE

1. USERS

1.1 DROP TABLE - USERS

Será la última tabla que borremos ya que hay muchas foreign keys tanto en el apartado de **HEALTH CARE**, **CONSPIRATIONS** o **REAL STATE** que relacionan diferentes tablas con la llave foránea id_user de la tabla users. Por lo tanto para evitar errores de orfandad (si borraremos primero users, esas tablas nunca podrían relacionar la llave foránea id_user con la tabla users).

Código:

```
DROP TABLE IF EXISTS users;
```

1.2 CREATE TABLE - USERS

Esta tabla es la más importante de toda la base de datos porque se utilizará en todas las secciones.

```
MariaDB [evilCorp]> DESCRIBE users;
+----------+---------------------+------+-----+---------+----------------+
| Field    | Type                | Null | Key | Default | Extra          |
+----------+---------------------+------+-----+---------+----------------+
| id_user  | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| username | varchar(64)         | NO   |     | NULL    |                |
| password | varchar(32)         | NO   |     | NULL    |                |
| email    | varchar(64)         | NO   |     | NULL    |                |
| name     | varchar(32)         | NO   |     | NULL    |                |
| dead     | tinyint(1)          | YES  |     | 0       |                |
+----------+---------------------+------+-----+---------+----------------+
```

Código:

```
CREATE TABLE users(
        id_user BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
        username VARCHAR(64) NOT NULL,
        password VARCHAR(32) NOT NULL,
        email VARCHAR(64) NOT NULL,
        name VARCHAR(32) NOT NULL
);
```

1.3 INSERT INTO - USERS

```
INSERT INTO users(username, password, email, name)
VALUES
        ('root', 'admin123', 'root@enti.cat', 'Admin'),
        ('marcel8', 'marcelito123', 'marcel.seto@enti.cat', 'Marcel'),
        ('pablo21', 'pablitosocias', 'pablo.lopez@enti.cat', 'Pablo'),
        ('johnK', 'johnasa12', 'john.largao@enti.cat', 'John'),
        ('guillermojazz', 'saxofon21', 'guillem.agutllo@enti.cat', 'Guillem'),
        ('juanramon12', 'juanrapapu', 'juan.ramon@enti.cat', 'Juan'),
        ('franciscana09','francisca12', 'francisca.moya@enti.cat', 'Francisca'),
        ('micaela4', 'micapica', 'micaela.papu@enti.cat', 'Micaela'),
        ('estefi45', 'estefipiti', 'estefania.garfia@enti.cat', 'Estefania');
```

1.4 OTROS SCRIPTS - USERS

1.4.1 ALTER USERS TABLE

Esta cláusula ALTER TABLE se utiliza para agregar una nueva columna llamada dead a la tabla users sin tener que borrar y volver a escribir la tabla users de nuestra base de datos evilCorp. La columna tiene el tipo de datos BOOLEAN, lo que significa que solo podrá contener dos valores: TRUE o FALSE.
La cláusula DEFAULT FALSE establece el valor predeterminado de la columna en FALSE. Esto significa que cuando se inserta un nuevo registro en la tabla sin especificar un valor para la columna dead, se asignará automáticamente el valor en FALSE. Además si ya hay usuarios creados, se les añadirá automáticamente la columna dead con el valor por defecto FALSE.

Código:

```
ALTER TABLE users
ADD COLUMN dead BOOLEAN DEFAULT FALSE;
```

1.4.2 KILL USERS

Este procedimiento llamado kill_user tiene como objetivo cambiar el estado de un usuario de la tabla users para indicar que está muerto (dead = true). El procedimiento acepta un parámetro de entrada en este caso _name_user (VARCHAR(64)).

Empezamos declarando variables locales entre ellas: user_dead, la cuál nos sirve para almacenar el estado actual de vida del usuario y también tenemos la variable is_user_alive que nos srive para almacenar un valor booleano que nos indica si el usuario está vivo o no.
A continuación realizamos una consulta SELECT para obtener el valor de la columna dead(estado de vida) de la tabla users para el usuario cuyo nombre coincida con _name_user (la variable que pasamos por parámetro al inicio del procedure). El resultado de esta consulta se lo asignamos a la variable is_user_alive.

Después, se utiliza una estructura IF para comprobar el estado del usuario. Si el usuario está muerto, se devuelve un mensaje indicando que el usuario ya está muerto. En caso contrario, se ejecuta una instrucción UPDATE para modificar la columna "dead" del usuario en la tabla "users" y establecerla en TRUE (indicando que el usuario está muerto). Este update lo realizamo para el usuario cuyo nombre coincide con _name_user. Después de actualizar el estado del usuario, devolvemos un mensaje indicando que el usuario ha sido asesinado.

Código:

```
DROP PROCEDURE IF EXISTS kill_user;

DELIMITER $$

CREATE PROCEDURE kill_user(IN _name_user VARCHAR(64))
BEGIN
	DECLARE user_dead VARCHAR(32);
	DECLARE is_user_alive BOOLEAN;

	SELECT users.dead INTO is_user_alive
	FROM users
	WHERE users.name = _name_user;

	IF is_user_alive THEN

	SELECT CONCAT('El usuario con nombre ', _name_user, ' está muerte.') AS
message;

	ELSE

	UPDATE users
```

```
        SET users.dead = TRUE
        WHERE users.name = _name_user;

        SELECT CONCAT('He matado a ', _name_user, '. JIJIJIJI!') AS message;
        END IF;

END $$

DELIMITER ;
```

Comprobación:

El único usuario de la tabla users que está muerto es Pablo, probamos con él el procedimiento kill_user y como podemos ver al estar muerto nos muestra un mensaje de que Pablo ya está muerto.
Al usar el procedimiento con Estefania, éste nos indica con un mensaje que la hemos matado, además de actualizar el estado a true en la columna dead de la tabla users.

```
MariaDB [evilCorp]> SELECT * FROM users;
+---------+--------------+--------------+--------------------------+-----------+------+
| id_user | username     | password     | email                    | name      | dead |
+---------+--------------+--------------+--------------------------+-----------+------+
|       1 | root         | admin123     | root@enti.cat            | Admin     |    0 |
|       2 | marcel8      | marcelito123 | marcel.seto@enti.cat     | Marcel    |    0 |
|       3 | pablo21      | pablitosocias| pablo.lopez@enti.cat     | Pablo     |    1 |
|       4 | johnK        | johnasa12    | john.largao@enti.cat     | John      |    0 |
|       5 | guillermojazz| saxofon21    | guillem.agutllo@enti.cat | Guillem   |    0 |
|       6 | juanramon12  | juanrapapu   | juan.ramon@enti.cat      | Juan      |    0 |
|       7 | franciscana09| francisca12  | francisca.moya@enti.cat  | Francisca |    0 |
|       8 | micaela4     | micapica     | micaela.papu@enti.cat    | Micaela   |    0 |
|       9 | estefi45     | estefipiti   | estefania.garfia@enti.cat| Estefania |    0 |
+---------+--------------+--------------+--------------------------+-----------+------+
```

```
MariaDB [evilCorp]> CALL kill_user("Pablo");
+-----------------------------------------+
| message                                 |
+-----------------------------------------+
| El usuario con nombre Pablo está muerte.  |
+-----------------------------------------+
```

```
MariaDB [evilCorp]> CALL kill_user("Estefania");
+-----------------------------------+
| message                           |
+-----------------------------------+
| He matado a Estefania. JIJIJIJI!  |
+-----------------------------------+
```

```
MariaDB [evilCorp]> SELECT * FROM users;
+---------+--------------+--------------+--------------------------+-----------+------+
| id_user | username     | password     | email                    | name      | dead |
+---------+--------------+--------------+--------------------------+-----------+------+
|       1 | root         | admin123     | root@enti.cat            | Admin     |    0 |
|       2 | marcel8      | marcelito123 | marcel.seto@enti.cat     | Marcel    |    0 |
|       3 | pablo21      | pablitosocias| pablo.lopez@enti.cat     | Pablo     |    1 |
|       4 | johnK        | johnasa12    | john.largao@enti.cat     | John      |    0 |
|       5 | guillermojazz| saxofon21    | guillem.agutllo@enti.cat | Guillem   |    0 |
|       6 | juanramon12  | juanrapapu   | juan.ramon@enti.cat      | Juan      |    0 |
|       7 | franciscana09| francisca12  | francisca.moya@enti.cat  | Francisca |    0 |
|       8 | micaela4     | micapica     | micaela.papu@enti.cat    | Micaela   |    0 |
|       9 | estefi45     | estefipiti   | estefania.garfia@enti.cat| Estefania |    1 |
+---------+--------------+--------------+--------------------------+-----------+------+
```

---

2. HEALTH CARE

2.1 DROP TABLES - HEALTH CARE

Primero decir que las tablas con las que trataremos a continuación no están relacionadas con las tablas tanto del apartado **CONSPIRATIONS** como del apartado **REAL STATE**. En cambio como he dicho anteriormente si que hay referencias a la tabla **users**.
Importante borrar primero la tabla treatments ya que esta contiene referencias (llaves foráneas) de las tablas diagnoses y medicines. Una vez borramos treatments la siguiente será diagnoses ya que contiene referencias tanto de users, diseases y doctors. A continuación borraremos las tablas diseases, medicines y doctors sin importar el orden en las que las borremos ya que no contienen referencias a ninguna otra tabla.

Código:

```
DROP TABLE IF EXISTS treatments;
DROP TABLE IF EXISTS diagnoses;
DROP TABLE IF EXISTS diseases;
DROP TABLE IF EXISTS medicines;
DROP TABLE IF EXISTS doctors;
```

2.2 CREATE TABLES - HEALTH CARE

MEDICINES: En esta tabla se almacenarán las distintas medicinas que tendrá EvilCorp a su disposición.

```
MariaDB [evilCorp]> DESCRIBE medicines;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_medicine | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| medicine    | varchar(192)     | NO   |     | NULL    |                |
| cost        | decimal(9,2)     | NO   |     | NULL    |                |
| price       | decimal(9,2)     | NO   |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
```

DISEASES: En esta tabla se almacenarán las distintas enfermedades que puedan existir o las que más interesen a EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE diseases;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_disease  | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| disease     | varchar(64)      | NO   |     | NULL    |                |
| symptoms    | text             | NO   |     | NULL    |                |
| description | text             | NO   |     | NULL    |                |
| deadly      | tinyint(1)       | NO   |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
```

TREATMENTS: Esta tabla la usaremos para juntar una medicina en concreto y un diagnóstico.

```
MariaDB [evilCorp]> DESCRIBE treatments;
+--------------+---------------------+------+-----+---------+----------------+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------+
| id_treatment | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| id_diagnosis | bigint(20) unsigned | NO   | MUL | NULL    |                |
| id_medicine  | int(10) unsigned    | NO   | MUL | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
```

DOCTORS: Esta tabla es donde se guardaran los datos de los doctores que están a disposición de EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE doctors;
+-----------+------------------+------+-----+---------+----------------+
| Field     | Type             | Null | Key | Default | Extra          |
+-----------+------------------+------+-----+---------+----------------+
| id_doctor | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| doctor    | varchar(64)      | YES  |     | NULL    |                |
+-----------+------------------+------+-----+---------+----------------+
```

DIAGNOSES: Esta tabla se encarga de almacenar los datos de un diagnóstico completo.

```
MariaDB [evilCorp]> DESCRIBE diagnoses;
+--------------+---------------------+------+-----+---------+----------------+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------+
| id_diagnosis | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| diagnosis    | text                | NO   |     | NULL    |                |
| id_user      | bigint(20) unsigned | NO   | MUL | NULL    |                |
| id_disease   | int(10) unsigned    | NO   | MUL | NULL    |                |
| id_doctor    | int(10) unsigned    | NO   | MUL | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
```

Los CREATES se realizan en el orden contrario de los DROPS. Para no generar errores con la creación de llaves foráneas y referenciaciones de tablas.

Código:

```
CREATE TABLE diseases(
	id_disease INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	disease VARCHAR(64) NOT NULL,
	symptoms TEXT NOT NULL,
	description TEXT NOT NULL,
	deadly BOOLEAN NOT NULL
);

CREATE TABLE medicines(
	id_medicine INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	medicine VARCHAR(192) NOT NULL,
	cost DECIMAL(9, 2) NOT NULL,
	price DECIMAL(9, 2) NOT NULL
);

CREATE TABLE doctors(
```

```
	id_doctor INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	doctor VARCHAR(64)
);

CREATE TABLE diagnoses(
	id_diagnosis BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	diagnosis TEXT NOT NULL,
	id_user BIGINT UNSIGNED NOT NULL,
	id_disease INT UNSIGNED NOT NULL,
	id_doctor INT UNSIGNED NOT NULL,
	FOREIGN KEY(id_user) REFERENCES users(id_user),
	FOREIGN KEY(id_disease) REFERENCES diseases(id_disease),
	FOREIGN KEY(id_doctor) REFERENCES doctors(id_doctor)
);

CREATE TABLE treatments(
	id_treatment BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	id_diagnosis BIGINT UNSIGNED NOT NULL,
	id_medicine INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_diagnosis) REFERENCES diagnoses(id_diagnosis),
	FOREIGN KEY (id_medicine) REFERENCES medicines(id_medicine)
);
```

2.3 INSERTS INTO - HEALTH CARE

Los **INSERTS** de **HEALTH CARE** són los siguientes:

```
INSERT INTO diseases(disease, symptoms, description, deadly)
VALUES
	('Dolor de cabeza', 'Fuerte dolor de cabeza acompañado de secrecion nasal', 'Mucho dolor de cabeza en todos lados', false),
	('Ebola', 'Transformación en mono', 'Sale mucha sangre de todos lados', true),
	('Covid-19', 'Mocos y tos seca', 'Tienes que estarte sonando constantemente y aislado', true),
	('Cancer', 'Perdida de peso y malestar', 'Crecimiento de celulas muertas', true),
	('Diabetes', 'Perdida de apetito, cansancio constante...', 'Falta de azucar en la sangre', false);

INSERT INTO medicines(medicine, cost, price)
VALUES
	('Ibuprofeno', 4.50, 9.05),
```

```
        ('Gamfos', 20.20, 38.99),
        ('Sangtness', 12.43, 37.55),
        ('Rinomer', 2.12, 5.50),
        ('Glucosan', 6.37, 15.98);

INSERT INTO doctors(doctor)
VALUES
        ('Juan Ramón Salado'),
        ('Pablo Lopez Socias'),
        ('Sofia Viralta Lambón'),
        ('Juana Chacón Lopez'),
        ('Roberta Miguelez Beltrán');

INSERT INTO diagnoses(diagnosis, id_user, id_disease, id_doctor)
VALUES
        ('El paciente padece de Covid-19 sebero', 2, 3, 1),
        ('El paciente tiene Ebola de grado 1', 1, 2, 2),
        ('El paciente tiene Cancer de Colon en fase no avanzada', 4, 4, 3),
        ('El paciente tiene una cefalea que le impide trabajar', 5, 1, 4),
        ('El paciente presenta una diabetes de tipo 1 en nivel avanzado', 3, 5, 5);
INSERT INTO treatments(id_diagnosis, id_medicine)
VALUES
        (1, 4), (2, 3), (3, 2), (4, 1), (4, 4), (5, 5);
```

2.4 OTROS SCRIPTS - HEALTH CARE

2.4.1 MEDICINES COST

Esta view llamada medicineplanet_money_revenue tiene como objetivo mostrar los ingresos totales que representan las medicinas en cada planeta.

La vista la creamos a través de una consulta SELECT que combina varias tablas de la base de datos, entre ellas: users, medicines, diagnoses, treatments, planets y users_planets.

Vamos haciendo la unión entre las tablas a través de la cláusula WHERE para asegurarnos que los registros estén correctamente relacionados. Comparamos el ID del planeta en la tabla planets con el ID del planeta en la tabla users_planets. Además también comparamos los IDs de usuario, diagnóstico y tratamiento para asegurar que la relación sea la adecuada para poder conseguir el objetivo final del procedure.

Utilizamos la función SUM para calcular la suma total de los precios de los medicamentos asociados a los diagnósticos y tratamientos correspondientes. Y finalmente con la cláusula GROUP BY agrupamos los resultados de manera DESC (descendente) por el nombre del planeta (planets.planet), lo que significa que los resultados mostrarán los ingresos totales que generan las medicinas para cada planeta.

Código:

```
DROP VIEW IF EXISTS medicineplanet_money_revenue;

CREATE VIEW medicineplanet_money_revenue
AS SELECT planets.planet, SUM(price) revenue
FROM users, medicines, diagnoses, treatments, planets, users_planets
WHERE planets.id_planet = users_planets.id_planet
AND diagnoses.id_user = users_planets.id_user
AND diagnoses.id_diagnosis=treatments.id_diagnosis
AND medicines.id_medicine = treatments.id_medicine
GROUP BY planets.planet
ORDER BY revenue DESC;
```

Comprobación:

```
MariaDB [evilCorp]> select * from medicines;
+-------------+------------+-------+-------+
| id_medicine | medicine   | cost  | price |
+-------------+------------+-------+-------+
|           1 | Ibuprofeno |  4.50 |  9.05 |
|           2 | Gamfos     | 20.20 | 38.99 |
|           3 | Sangtness  | 12.43 | 37.55 |
|           4 | Rinomer    |  2.12 |  5.50 |
|           5 | Glucosan   |  6.37 | 15.98 |
+-------------+------------+-------+-------+
```

```
MariaDB [evilCorp]> select * from diseases;
+------------+-----------------+-------------------------------------------------------------+-----------
-----------------------------------------+--------+
| id_disease | disease         | symptoms                                                    | description
                                         | deadly |
+------------+-----------------+-------------------------------------------------------------+-----------
-----------------------------------------+--------+
|          1 | Dolor de cabeza | Fuerte dolor de cabeza acompañado de secrecion nasal        | Mucho dolor
 de cabeza en todos lados                |      0 |
|          2 | Ebola           | Transformación en mono                                      | Sale mucha
sangre de todos lados                    |      1 |
|          3 | Covid-19        | Mocos y tos seca                                            | Tienes que
estarte sonando constantemente y aislado |      1 |
|          4 | Cancer          | Perdida de peso y malestar                                  | Crecimiento
 de celulas muertas                      |      1 |
|          5 | Diabetes        | Perdida de apetito, cansancio constante...                  | Falta de az
ucar en la sangre                        |      0 |
+------------+-----------------+-------------------------------------------------------------+-----------
-----------------------------------------+--------+
```

```
MariaDB [evilCorp]> select * from users;
+---------+--------------+--------------+---------------------------+-----------+------+
| id_user | username     | password     | email                     | name      | dead |
+---------+--------------+--------------+---------------------------+-----------+------+
|       1 | root         | admin123     | root@enti.cat             | Admin     |    0 |
|       2 | marcel8      | marcelito123 | marcel.seto@enti.cat      | Marcel    |    0 |
|       3 | pablo21      | pablitosocias | pablo.lopez@enti.cat     | Pablo     |    1 |
|       4 | johnK        | johnasa12    | john.largao@enti.cat      | John      |    0 |
|       5 | guillermojazz | saxofon21   | guillem.agutllo@enti.cat  | Guillem   |    0 |
|       6 | juanramon12  | juanrapapu   | juan.ramon@enti.cat       | Juan      |    0 |
|       7 | franciscana09 | francisca12 | francisca.moya@enti.cat   | Francisca |    0 |
|       8 | micaela4     | micapica     | micaela.papu@enti.cat     | Micaela   |    0 |
|       9 | estefi45     | estefipiti   | estefania.garfia@enti.cat | Estefania |    1 |
+---------+--------------+--------------+---------------------------+-----------+------+
```

```
MariaDB [evilCorp]> select * from diagnoses;
+--------------+-----------------------------------------------------------------+---------+----------
--+-----------+
| id_diagnosis | diagnosis                                                       | id_user | id_diseas
e | id_doctor |
+--------------+-----------------------------------------------------------------+---------+----------
--+-----------+
|            1 | El paciente padece de Covid-19 sebero                          |       2 |
3 |         1 |
|            2 | El paciente tiene Ebola de grado 1                             |       1 |
2 |         2 |
|            3 | El paciente tiene Cancer de Colon en fase no avanzada          |       4 |
4 |         3 |
|            4 | El paciente tiene una cefalea que le impide trabajar           |       5 |
1 |         4 |
|            5 | El paciente presenta una diabetes de tipo 1 en nivel avanzado  |       3 |
5 |         5 |
+--------------+-----------------------------------------------------------------+---------+----------
--+-----------+
```

```
MariaDB [evilCorp]> select * from treatments;
+--------------+--------------+-------------+
| id_treatment | id_diagnosis | id_medicine |
+--------------+--------------+-------------+
|            1 |            1 |           4 |
|            2 |            2 |           3 |
|            3 |            3 |           2 |
|            4 |            4 |           1 |
|            5 |            4 |           4 |
|            6 |            5 |           5 |
+--------------+--------------+-------------+
```

```
MariaDB [evilCorp]> SELECT * FROM medicineplanet_money_revenue;
+---------+---------+
| planet  | revenue |
+---------+---------+
| Marte   |  350.91 |
| Tierra  |  337.95 |
| Jupiter |  143.82 |
| Pluton  |  130.95 |
| Venus   |   49.50 |
+---------+---------+
```

---

3. CONSPIRATIONS

3.1 DROP TABLES - CONSPIRATIONS

Primero decir que las tablas con las que trataremos a continuación no están relacionadas con las tablas tanto del apartado **HEALTH CARE** como del apartado **REAL STATE**. En cambio como he dicho anteriormente si que hay referencias a la tabla **users**.

En este apartado es importante borrar primero la tabla users_conspirations ya que contiene referencias tanto de la tabla users como de la de conspirations. Además la tabla illuminatis también debemos borrarla en este caso antes de users ya que illuminatis hace referencia a id_user.

Código:

```
DROP TABLE IF EXISTS users_adresses;
```

```
DROP TABLE IF EXISTS users_planets;
DROP TABLE IF EXISTS adresses;
DROP TABLE IF EXISTS streets;
DROP TABLE IF EXISTS cities;
DROP TABLE IF EXISTS countries;
DROP TABLE IF EXISTS street_numbers;
DROP TABLE IF EXISTS staircases;
DROP TABLE IF EXISTS floors;
DROP TABLE IF EXISTS doors;
DROP TABLE IF EXISTS zip_codes;
DROP TABLE IF EXISTS planets;
```

3.2 CREATE TABLES - CONSPIRATIONS

ILUMINATIS: En esta tabla solo se almacenarán los usuarios que son illuminatis.

```
MariaDB [evilCorp]> DESCRIBE iluminatis;
+--------------+---------------------+------+-----+---------+----------------+
| Field        | Type                | Null | Key | Default | Extra          |
+--------------+---------------------+------+-----+---------+----------------+
| id_iluminati | int(10) unsigned    | NO   | PRI | NULL    | auto_increment |
| id_user      | bigint(20) unsigned | NO   | MUL | NULL    |                |
+--------------+---------------------+------+-----+---------+----------------+
```

CONSPIRATIONS: En esta tabla guardaremos las conspiraciones que hay hoy en día.

```
MariaDB [evilCorp]> DESCRIBE conspirations;
+-----------------+------------------+------+-----+---------+----------------+
| Field           | Type             | Null | Key | Default | Extra          |
+-----------------+------------------+------+-----+---------+----------------+
| id_conspiration | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| conspiration    | varchar(32)      | NO   |     | NULL    |                |
| description     | text             | NO   |     | NULL    |                |
+-----------------+------------------+------+-----+---------+----------------+
```

USERS_CONSPIRATIONS: Esta tabla almacenará las conspiraciones y los usuarios que creen en ellas.

```
MariaDB [evilCorp]> DESCRIBE users_conspirations;
+--------------------+---------------------+------+-----+---------+----------------+
| Field              | Type                | Null | Key | Default | Extra          |
+--------------------+---------------------+------+-----+---------+----------------+
| id_user_conspiration | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| id_user            | bigint(20) unsigned | NO   | MUL | NULL    |                |
| id_conspiration    | int(10) unsigned    | NO   | MUL | NULL    |                |
+--------------------+---------------------+------+-----+---------+----------------+
```

Los CREATES se realizan en el orden contrario de los DROPS. Para no generar errores con la creación de llaves foráneas y referenciaciones de tablas.

Código:

```
CREATE TABLE conspirations(
	id_conspiration INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	conspiration VARCHAR(32) NOT NULL,
	description TEXT NOT NULL
);

CREATE TABLE iluminatis(
	id_iluminati INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	id_user BIGINT UNSIGNED NOT NULL,
	FOREIGN KEY (id_user) REFERENCES users(id_user)
);

CREATE TABLE users_conspirations(
	id_user_conspiration BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY
KEY,
	id_user BIGINT UNSIGNED NOT NULL,
	id_conspiration INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_user) REFERENCES users(id_user),
	FOREIGN KEY (id_conspiration) REFERENCES conspirations(id_conspiration)
);
```

3.3 INSERTS INTO - CONSPIRATIONS

Los **INSERTS** de **CONSPIRATIONS** són los siguientes:

```
INSERT INTO conspirations(conspiration, description)
VALUES
	('Tierraplanista', 'Consiste en creer que la tierra no es redonda, sino que es
plana'),
	('Hitler vivo y coleando', 'Consiste en creer que Hitler no murió en el final de la
segunda guerra mundial'),
	('Efecto Mandela', 'Consiste en creer que pikachu tiene la cola negra'),
	('Iluminismo', 'Consiste en creer que los iluminatis existen'),
	('Beticismo', 'Consiste en creer que el betis puede ganar algo');

INSERT INTO iluminatis(id_user)
```

```
VALUES
        (1), (4);

INSERT INTO users_conspirations(id_user, id_conspiration)
VALUES
        (1, 5), (1, 2), (3, 5), (4, 3), (4, 1);
```

---

4. REAL STATE

4.1 DROP TABLES - REAL STATE

Primero decir que las tablas con las que trataremos a continuación no están relacionadas con las tablas tanto del apartado **HEALTH CARE** como del apartado **CONSPIRATIONS**. En cambio como he dicho anteriormente si que hay referencias a la tabla **users**.

En este apartado es importante borrar primero la tabla users_adresses y users_planets ya que contiene referencias tanto de la tabla users como de adresses, como de planets. A continuación borramos adresses, ya contiene referencias de una gran cantidad de tablas (streets, street_numbers, staircases, floors, doors, zip_codes). Además tenemos que borrar en el siguiente orden 4 tablas que contienen 1 llave foránea de la tabla que hay que borrar a continuación (streets (llave foránea de cities), cities (llave foránea de countries), countries (llave foránea de planets) y planets)

Código:

```
DROP TABLE IF EXISTS users_adresses;
DROP TABLE IF EXISTS users_planets;
DROP TABLE IF EXISTS adresses;
DROP TABLE IF EXISTS streets;
DROP TABLE IF EXISTS cities;
DROP TABLE IF EXISTS countries;
DROP TABLE IF EXISTS street_numbers;
DROP TABLE IF EXISTS staircases;
DROP TABLE IF EXISTS floors;
DROP TABLE IF EXISTS doors;
DROP TABLE IF EXISTS zip_codes;
DROP TABLE IF EXISTS planets;
```

4.2 CREATE TABLES - REAL STATE

PLANETS: En esta tabla se almacenan los planetas que hay a disposición de EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE planets;
+-----------+------------------+------+-----+---------+----------------+
| Field     | Type             | Null | Key | Default | Extra          |
+-----------+------------------+------+-----+---------+----------------+
| id_planet | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| planet    | varchar(64)      | YES  |     | NULL    |                |
+-----------+------------------+------+-----+---------+----------------+
```

COUNTRIES: Este campo almacena los países de todos los planetas de EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE countries;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| id_country | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| country    | varchar(64)      | YES  |     | NULL    |                |
| id_planet  | int(10) unsigned | NO   | MUL | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
```

CITIES: Este campo almacena las ciudades de todos los paises de EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE cities;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| id_city    | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| city       | varchar(64)      | YES  |     | NULL    |                |
| id_country | int(10) unsigned | NO   | MUL | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
```

STREETS: Este campo almacena todas las calles de las diferentes ciudades de EvilCorp.

```
MariaDB [evilCorp]> DESCRIBE streets;
+-----------+---------------------+------+-----+---------+----------------+
| Field     | Type                | Null | Key | Default | Extra          |
+-----------+---------------------+------+-----+---------+----------------+
| id_street | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| street    | varchar(128)        | NO   |     | NULL    |                |
| id_city   | int(10) unsigned    | NO   | MUL | NULL    |                |
+-----------+---------------------+------+-----+---------+----------------+
```

STREET_NUMBERS: Esta tabla almacena los números de las calles.

```
MariaDB [evilCorp]> DESCRIBE street_numbers;
+------------------+------------------+------+-----+---------+----------------+
| Field            | Type             | Null | Key | Default | Extra          |
+------------------+------------------+------+-----+---------+----------------+
| id_street_number | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| street_number    | varchar(8)       | YES  |     | NULL    |                |
+------------------+------------------+------+-----+---------+----------------+
```

STAIRCASES: Esta tabla almacena los números de escalera.

```
MariaDB [evilCorp]> DESCRIBE staircases;
+--------------+------------------+------+-----+---------+----------------+
| Field        | Type             | Null | Key | Default | Extra          |
+--------------+------------------+------+-----+---------+----------------+
| id_staircase | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| staircase    | varchar(8)       | YES  |     | NULL    |                |
+--------------+------------------+------+-----+---------+----------------+
```

FLOORS: Esta tabla almacena los pisos de los edificios.

```
MariaDB [evilCorp]> DESCRIBE floors;
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
+----------+------------------+------+-----+---------+----------------+
| id_floor | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| floor    | varchar(8)       | YES  |     | NULL    |                |
+----------+------------------+------+-----+---------+----------------+
```

DOORS: Esta tabla almacena todas los números de puerta.

```
MariaDB [evilCorp]> DESCRIBE doors;
+---------+------------------+------+-----+---------+----------------+
| Field   | Type             | Null | Key | Default | Extra          |
+---------+------------------+------+-----+---------+----------------+
| id_door | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| door    | varchar(8)       | YES  |     | NULL    |                |
+---------+------------------+------+-----+---------+----------------+
```

ZIP_CODES: Esta tabla almacena todos los códigos postales posibles.

```
MariaDB [evilCorp]> DESCRIBE zip_codes;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id_zip_code | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| zip_code    | varchar(16)      | YES  |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
```

ADRESSES: Esta tabla almacena el conjunto de calle, número de calle, piso, puerta y código postal.

```
MariaDB [evilCorp]> DESCRIBE adresses;
+------------------+---------------------+------+-----+---------+----------------+
| Field            | Type                | Null | Key | Default | Extra          |
+------------------+---------------------+------+-----+---------+----------------+
| id_adress        | bigint(20) unsigned | NO   | PRI | NULL    | auto_increment |
| id_street        | bigint(20) unsigned | NO   | MUL | NULL    |                |
| id_street_number | int(10) unsigned    | NO   | MUL | NULL    |                |
| id_staircase     | int(10) unsigned    | NO   | MUL | NULL    |                |
| id_floor         | int(10) unsigned    | NO   | MUL | NULL    |                |
| id_door          | int(10) unsigned    | NO   | MUL | NULL    |                |
| id_zip_code      | int(10) unsigned    | NO   | MUL | NULL    |                |
+------------------+---------------------+------+-----+---------+----------------+
```

Los CREATES se realizan en el orden contrario de los DROPS. Para no generar errores con la creación de llaves foráneas y referenciaciones de tablas.

Código:

```
CREATE TABLE planets(
        id_planet INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
        planet VARCHAR(64)
);

CREATE TABLE countries(
        id_country INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
        country VARCHAR(64),
        id_planet INT UNSIGNED NOT NULL,
        FOREIGN KEY (id_planet) REFERENCES planets(id_planet)
);

CREATE TABLE cities(
        id_city INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
```

```
	city VARCHAR(64),
	id_country INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_country) REFERENCES countries(id_country)
);

CREATE TABLE streets(
	id_street BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	street VARCHAR(128) NOT NULL,
	id_city INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_city) REFERENCES cities(id_city)
);

CREATE TABLE street_numbers(
	id_street_number INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	street_number VARCHAR(8)
);

CREATE TABLE staircases(
	id_staircase INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	staircase VARCHAR(8)
);

CREATE TABLE floors(
	id_floor INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	`floor` VARCHAR(8)
);

CREATE TABLE doors(
	id_door INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	door VARCHAR(8)
);

CREATE TABLE zip_codes(
	id_zip_code INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	zip_code VARCHAR(16)
);

CREATE TABLE adresses(
	id_adress BIGINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
	id_street BIGINT UNSIGNED NOT NULL,
```

```
	id_street_number INT UNSIGNED NOT NULL,
	id_staircase INT UNSIGNED NOT NULL,
	id_floor INT UNSIGNED NOT NULL,
	id_door INT UNSIGNED NOT NULL,
	id_zip_code INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_street) REFERENCES streets(id_street),
	FOREIGN KEY (id_street_number) REFERENCES
street_numbers(id_street_number),
	FOREIGN KEY (id_staircase) REFERENCES staircases(id_staircase),
	FOREIGN KEY (id_floor) REFERENCES floors(id_floor),
	FOREIGN KEY (id_door) REFERENCES doors(id_door),
	FOREIGN KEY (id_zip_code) REFERENCES zip_codes(id_zip_code)
);

CREATE TABLE users_adresses (
	id_user_address BIGINT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
	id_user BIGINT UNSIGNED NOT NULL,
	id_adress BIGINT UNSIGNED NOT NULL,
	FOREIGN KEY (id_user) REFERENCES users(id_user),
	FOREIGN KEY (id_adress) REFERENCES adresses(id_adress)
);

CREATE TABLE users_planets (
	id_user_planet BIGINT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
	id_user BIGINT UNSIGNED NOT NULL,
	id_planet INT UNSIGNED NOT NULL,
	FOREIGN KEY (id_user) REFERENCES users(id_user),
	FOREIGN KEY (id_planet) REFERENCES planets(id_planet)
);
```

4.3 INSERTS INTO - REAL STATE

Los **INSERTS** de **REAL STATE** són los siguientes:

```
INSERT INTO planets(planet)
VALUES
	('Tierra'), ('Venus'), ('Jupiter'), ('Marte'), ('Pluton');

INSERT INTO countries(country, id_planet)
VALUES
	('España', 1),
```

```
        ('Colombia', 1),
        ('EEUU', 1),
        ('Wafe', 2),
        ('Venasaur', 2),
        ('Cangis', 2),
        ('Rar', 3),
        ('Osiris', 3),
        ('Redstone', 3),
        ('Canto', 4),
        ('Ristalk', 4),
        ('Mintol', 4),
        ('Kikiw', 5),
        ('Loic', 5),
        ('Promte', 5);

INSERT INTO cities(city, id_country)
VALUES
        ('Barcelona', 1),
        ('Waldrof', 4),
        ('Candor', 9),
        ('Elibol', 12),
        ('Bade', 14);

INSERT INTO streets(street, id_city)
VALUES
        ('Calle A', 1),
        ('Calle B', 2),
        ('Calle C', 3),
        ('Calle D', 4),
        ('Calle E', 5);

INSERT INTO street_numbers(street_number)
VALUES
        (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);

INSERT INTO staircases(staircase)
VALUES
        (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);

INSERT INTO floors(`floor`)
```

```
VALUES
        (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);

INSERT INTO doors(door)
VALUES
        (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);

INSERT INTO zip_codes(zip_code)
VALUES
        (1), (2), (3), (4), (5), (6), (7), (8), (9), (10);

INSERT INTO adresses(id_street, id_street_number, id_staircase, id_floor, id_door,
id_zip_code)
VALUES
        (1, 1, 1, 1, 1, 1),
        (2, 2, 2, 2, 2, 2),
        (3, 3, 3, 3, 3, 3),
        (4, 4, 4, 4, 4, 4),
        (5, 5, 5, 5, 5, 5);

INSERT INTO users_adresses(id_user, id_adress)
VALUES
        (1, 1), (2, 2), (3, 3), (4, 4), (5, 5);

INSERT INTO users_planets (id_user, id_planet)
VALUES
        (1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 4), (7, 1), (8, 4), (9, 5);
```

4.4 OTROS SCRIPTS

4.4.1 VIEW PERSONS X PLANET

La view llamada **count_persons_x_planet** tiene la intención de mostrar el recuento de personas / usuarios por planeta de nuestra base de datos EvilCorp.
La vista utiliza tres tablas: users, planets y users_planets las cuáles són seleccionadas con la cláusula FROM.
La consulta SELECT en la view realiza una combinación de las tres tablas utilizando las cláusulas WHERE para relacionar los usuarios con los planetas en los que se encuentran.

Luego, utilizamos la función de agregación COUNT para contar el número de registros que se dan en la tabla intermedia users_planets que relaciona a los usuarios con los planetas.
Después la cláusula GROUP BY agrupa los resultados por el nombre del planeta (planets.planet), lo que significa que se obtendrá el recuento de usuarios para cada planeta único en la base de datos. Finalmente, la vista se ordena en orden descendente (DESC) según el recuento anterior que hayamos hecho de los usuarios (count_user).

Código:

```
DROP VIEW IF EXISTS count_persons_x_planet;

CREATE VIEW count_persons_x_planet
AS SELECT planets.planet, COUNT(users_planets.id_user) AS count_user
FROM users, planets, users_planets
WHERE users.id_user = users_planets.id_user
AND planets.id_planet = users_planets.id_planet
GROUP BY planets.planet
ORDER BY count_user DESC;
```

Comprobación:

```
MariaDB [evilCorp]> SELECT * FROM users;
+---------+--------------+--------------+--------------------------+-----------+------+
| id_user | username     | password     | email                    | name      | dead |
+---------+--------------+--------------+--------------------------+-----------+------+
|       1 | root         | admin123     | root@enti.cat            | Admin     |    0 |
|       2 | marcel8      | marcelito123 | marcel.seto@enti.cat     | Marcel    |    0 |
|       3 | pablo21      | pablitosocias| pablo.lopez@enti.cat     | Pablo     |    1 |
|       4 | johnK        | johnasa12    | john.largao@enti.cat     | John      |    0 |
|       5 | guillermojazz| saxofon21    | guillem.agutllo@enti.cat | Guillem   |    0 |
|       6 | juanramon12  | juanrapapu   | juan.ramon@enti.cat      | Juan      |    0 |
|       7 | franciscana09| francisca12  | francisca.moya@enti.cat  | Francisca |    0 |
|       8 | micaela4     | micapica     | micaela.papu@enti.cat    | Micaela   |    0 |
|       9 | estefi45     | estefipiti   | estefania.garfia@enti.cat| Estefania |    1 |
+---------+--------------+--------------+--------------------------+-----------+------+
```

```
MariaDB [evilCorp]> SELECT * FROM planets;
+-----------+---------+
| id_planet | planet  |
+-----------+---------+
|         1 | Tierra  |
|         2 | Venus   |
|         3 | Jupiter |
|         4 | Marte   |
|         5 | Pluton  |
+-----------+---------+
```

```
MariaDB [evilCorp]> SELECT * FROM users_planets;
+----------------+---------+-----------+
| id_user_planet | id_user | id_planet |
+----------------+---------+-----------+
|              1 |       1 |         1 |
|              2 |       2 |         2 |
|              3 |       3 |         3 |
|              4 |       4 |         4 |
|              5 |       5 |         5 |
|              6 |       6 |         4 |
|              7 |       7 |         1 |
|              8 |       8 |         4 |
|              9 |       9 |         5 |
+----------------+---------+-----------+
```

```
MariaDB [evilCorp]> SELECT * FROM count_persons_x_planet;
+---------+------------+
| planet  | count_user |
+---------+------------+
| Marte   |          3 |
| Tierra  |          2 |
| Pluton  |          2 |
| Venus   |          1 |
| Jupiter |          1 |
+---------+------------+
```

4.4.2 FUNCTION RETURN RANDOM USER

Esta función llamada return_random_user retorna un usuario aleatorio de la tabla users que corresponda al planeta que le pasemos por parámetro. La función acepta un parámetro de entrada en este caso _id_planet, que se corresponda a un número entero unsigned.
Primero de todo comenzamos declarando variables locales entre ellas: users para almacenar el recuento de usuarios asociados con el planeta dado, random_num para almacenar un número aleatorio, random_user para almacenar el ID de usuario correspondiente al número aleatorio seleccionado y por último también declaramos la variable random_name para almacenar el nombre del usuario aleatorio seleccionado.

A continuación, asignamos el número de usuarios asociados con el planeta al que pertenece _id_planet (que pasamos por parámetro) a la variable users utilizando una consulta SELECT.
La variable random_num se le asigna un valor utilizando la función RAND(), que genera un número decimal aleatorio entre 0 y 1, y se multiplica por el recuento de usuarios que haya en ese momento. Luego, utilizamos la función FLOOR() para redondear el número hacia abajo y convertirlo en un número entero.
La variable random_user la establecemos seleccionando el ID de usuario de la tabla users_planets donde el ID de planeta coincide con _id_planet y se limita por el número aleatorio seleccionado utilizando la función LIMIT.

Finalmente, la variable random_name la establecemos seleccionando el nombre de usuario correspondiente al ID de usuario aleatorio de la tabla users que hayamos obtenido. La función devuelve el nombre de usuario aleatorio utilizando la instrucción RETURN.

Código:

```
DROP FUNCTION IF EXISTS return_random_user;

DELIMITER $$

CREATE FUNCTION return_random_user(_id_planet INT UNSIGNED) RETURNS
VARCHAR(32)
BEGIN

        DECLARE users INT;
        DECLARE random_num INT;
```

```
        DECLARE random_user INT;
        DECLARE random_name VARCHAR(32);

        SET users = (SELECT COUNT(*) FROM users_planets WHERE
users_planets.id_planet = _id_planet);
        SET random_num = FLOOR(RAND() * users);
        SET random_user = (SELECT id_user FROM users_planets WHERE
users_planets.id_planet = _id_planet LIMIT random_num, 1);
        SET random_name = (SELECT users.name FROM users WHERE users.id_user =
random_user);

        RETURN random_name;

END $$

DELIMITER ;
```

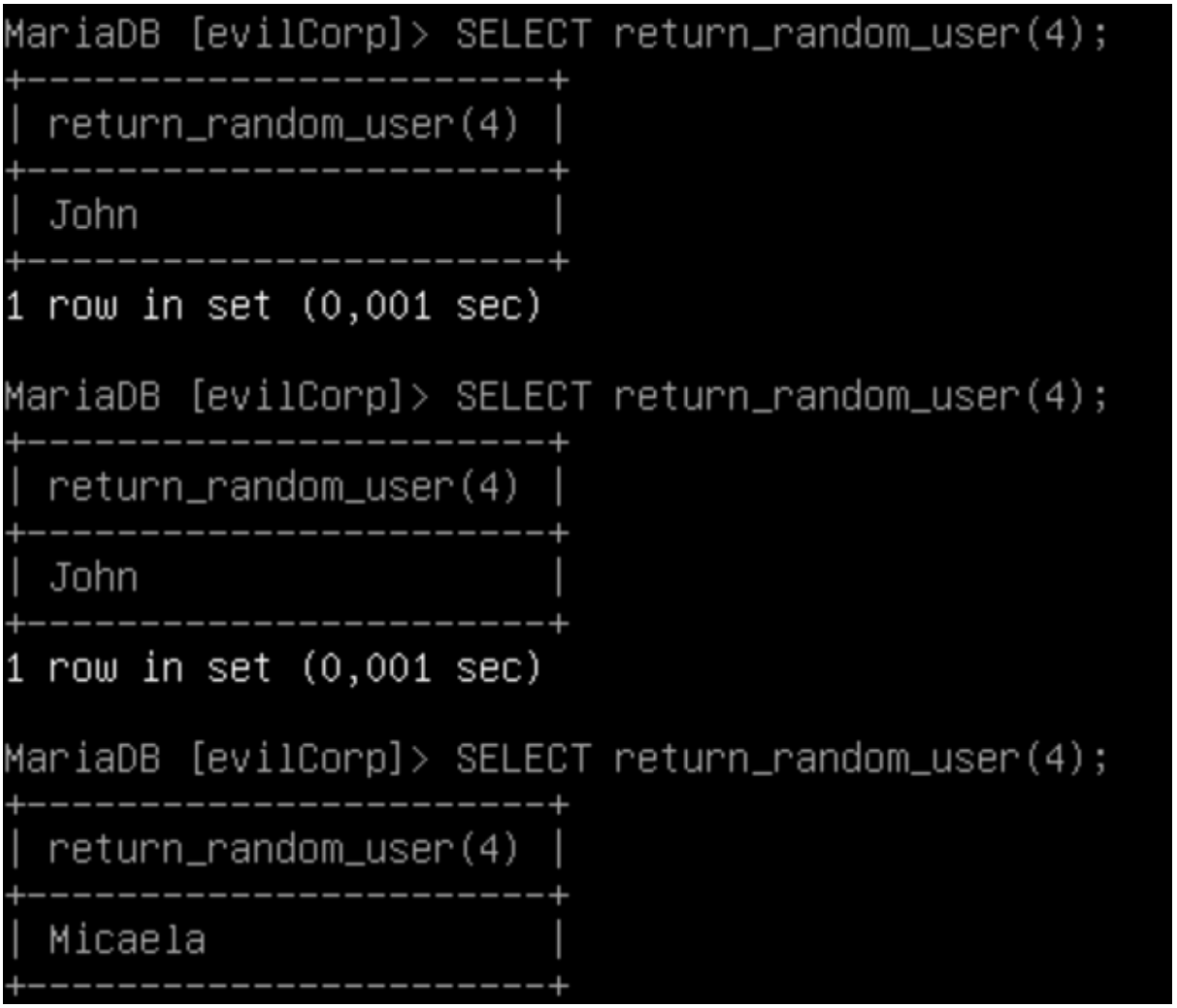
Comprobación:

En este caso para la comprobación, usaremos el planeta Marte con id = 4, que es la id que pasaremos por parámetro al llamar a la función. Por lo tanto solo nos puede dar los nombres: John, Juan y Micaela.

```
MariaDB [evilCorp]> SELECT * FROM users;
+---------+--------------+--------------+--------------------------+----------+------+
| id_user | username     | password     | email                    | name     | dead |
+---------+--------------+--------------+--------------------------+----------+------+
|       1 | root         | admin123     | root@enti.cat            | Admin    |    0 |
|       2 | marcel8      | marcelito123 | marcel.seto@enti.cat     | Marcel   |    0 |
|       3 | pablo21      | pablitosocias| pablo.lopez@enti.cat     | Pablo    |    1 |
|       4 | johnK        | johnasa12    | john.largao@enti.cat     | John     |    0 |
|       5 | guillermojazz| saxofon21    | guillem.agutllo@enti.cat | Guillem  |    0 |
|       6 | juanramon12  | juanrapapu   | juan.ramon@enti.cat      | Juan     |    0 |
|       7 | franciscana09| francisca12  | francisca.moya@enti.cat  | Francisca|    0 |
|       8 | micaela4     | micapica     | micaela.papu@enti.cat    | Micaela  |    0 |
|       9 | estefi45     | estefipiti   | estefania.garfia@enti.cat| Estefania|    1 |
+---------+--------------+--------------+--------------------------+----------+------+
```

```
MariaDB [evilCorp]> SELECT * FROM planets;
+-----------+---------+
| id_planet | planet  |
+-----------+---------+
|         1 | Tierra  |
|         2 | Venus   |
|         3 | Jupiter |
|         4 | Marte   |
|         5 | Pluton  |
+-----------+---------+
```

```
MariaDB [evilCorp]> SELECT * FROM users_planets;
+----------------+---------+-----------+
| id_user_planet | id_user | id_planet |
+----------------+---------+-----------+
|              1 |       1 |         1 |
|              2 |       2 |         2 |
|              3 |       3 |         3 |
|              4 |       4 |         4 |
|              5 |       5 |         5 |
|              6 |       6 |         4 |
|              7 |       7 |         1 |
|              8 |       8 |         4 |
|              9 |       9 |         5 |
+----------------+---------+-----------+
```



5. LA PARCA

La parca es un usuario de la base de datos evilCorp el cúal sólo pueda hacer select y update en la tabla de usuarios.

5.1 SCRIPT CREACIÓN + PRIVILEGIOS

DROP USER IF EXISTS 'parca'@'localhost';
Esta instrucción la utilizamos para eliminar un usuario de la base de datos. En este caso, se intenta eliminar el usuario llamado parca que contiene la ip del localhost (127.0.0.1). Con el IF EXISTS nos aseguramos de que no se produzca un error en el caso de que el usuario no exista.

CREATE USER 'parca'@'localhost' IDENTIFIED BY 'parca123';
Esta instrucción crea un nuevo usuario llamado parca con la ip del localhost. El cuál se identificará utilizando la contraseña 'parca123'.

GRANT SELECT, UPDATE ON evilCorp.users TO 'parca'@'localhost';
Esta instrucción la utilizamos para otorgar privilegios al usuario parca en la base de datos 'evilCorp' concretamente en la tabla users (por lo tanto el usuario solo tendrá permisos para realizar estas acciones en la tabla'users). En este caso, se le otorgan los privilegios de SELECT (lectura) y UPDATE (actualización).

FLUSH PRIVILEGES;
Finalmente hacemos un FLUSH PRIVILEGES lo cuál nos permite actualizar los privilegios de la base de datos después de realizar cambios en los usuarios y sus privilegios. Es imprescindible ejecutar esta instrucción para que los cambios sean actualizados.

Código:

```
DROP USER IF EXISTS 'parca'@'localhost';

CREATE USER 'parca'@'localhost' IDENTIFIED BY 'parca123';
GRANT SELECT, UPDATE ON evilCorp.users TO 'parca'@'localhost';

FLUSH PRIVILEGES;
```

5.2 EJECUCIÓN FUNCIONES

```
MariaDB [evilCorp]> SELECT return_random_user(4);
ERROR 1370 (42000): execute command denied to user 'parca'@'localhost' for routine 'evilCorp.return_
random_user'
```

```
MariaDB [evilCorp]> CALL kill_user("Pablo");
ERROR 1370 (42000): execute command denied to user 'parca'@'localhost' for routine 'evilCorp.kill_us
er'
```

Como podemos ver, con el usuario parca, no tenemos los permisos necesarios para ejecutar ninguno de estos dos métodos. Esto se debe a que en los permisos no le hemos puesto los permisos para hacer drop, ni los permisos en las tablas necesarias para sacar el usuario random.