Tarea S4.01. Creación de Base de Datos

Descripción

Según algunos archivos CSV, diseñará y creará su base de datos.

Descargue archivos CSV, estúdielos y diseñe una base de datos con un esquema estrella que contenga, al menos 4 tablas de las cuales puede realizar las siguientes consultas:

Al estudiar los CSV, me di cuenta que la estructura era similar a los datos que venimos trabajando, entonces empecé a crear una nueva base de datos, y sus tablas, para introducir los CSV.

Ahora antes de cargar los CSV, debo interiorizarme en qué tipos de datos estoy manipulando, y he visto que nos encontramos con la tabla transaction, company, credit_card(que la hemos creado antes), que tienen estructuras similares, pero no iguales.

He usado el Visual Studio Code para visualizar los CSV y una extension llamada rainbow para diferenciar los distintos campos y filas.
Las diferencias que he encontrado son:

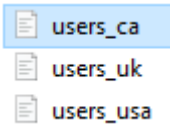
1. Dentro del CSV de “transaction”, company_id = business_id. Como estaba acostumbrado y además dentro de la tabla company también se encuentra el nombre anterior, voy a cambiarlo por company_id.

```
1   id;card_id;business_id;timestamp;amount;declined;product_ids;user_id;lat;longitude
```

2. Luego hay un nuevo campo que se llama products_id, que indicaran los productos que se han adquirido en una transaction. Están separados por comas. Por ejemplo: 22, 33, 11, dependiendo si han adquirido 1 o varios productos.

```
product_ids;97, 41, 3;
```

3. Ahora nos encontramos con 3 tablas de users, separados por país. Por Estados Unidos, Reino Unido y Canadá. Y a su vez nos provee de más información sobre cada usuario.




```
id,name,surname,phone,email,birth_date,country,city,postal_code,address
```

Luego de analizar los CSV, procederé a crear la base de datos.

```sql
-- Creo la base de datos
CREATE DATABASE transactions_2; -- Le voy a llamar a esta base de datos transactions_2
USE transactions_2; -- Indico que voy a usar esta base de datos
```

Una vez que tengo la base de datos creada, procedo a crear las tablas en cuestión para luego cargar los CSV correspondientes.

```sql
-- Creo la tabla transaction
CREATE TABLE IF NOT EXISTS transaction (
    id VARCHAR(255) NOT NULL PRIMARY KEY,
    credit_card VARCHAR(15) NULL,
    company_id VARCHAR(20) NULL,
    timestamp TIMESTAMP NULL,
    amount DECIMAL(10, 2) NULL,
    declined BOOLEAN NULL,
    product_ids VARCHAR(15) NULL,
    user_id INT NULL,
    lat FLOAT null,
    longitude FLOAT null

    );

CREATE TABLE IF NOT EXISTS company (
    id VARCHAR(15) PRIMARY KEY,
    company_name VARCHAR(255) ,
    phone VARCHAR(15) ,
    email VARCHAR(100) ,
    country VARCHAR(100) ,
    website VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS credit_card (
    id VARCHAR(100) PRIMARY KEY,
    user_id INT,
    iban VARCHAR(50) NOT NULL,
    pan VARCHAR(100),
    pin INT,
    cvv SMALLINT,
    track1 VARCHAR(255),
    track2 VARCHAR(255),
    expiring_date VARCHAR(10)
);
```

Luego creo las 3 tablas ‘user’ que tengo = (user_usa, user_uk, user_ca)

```sql
CREATE TABLE IF NOT EXISTS user_usa (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);
CREATE TABLE IF NOT EXISTS user_uk (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS user_ca (
    id VARCHAR(10) PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);
```

Intuyo que al tener 3 tablas de users con ids que están dentro de users_id en transaction, pero al mismo tiempo son partes de la totalidad de transaction, tendré que unir estos users en una sola tabla. Entonces creare una tabla users_all donde voy a introducir estos datos así podre relacionarla.

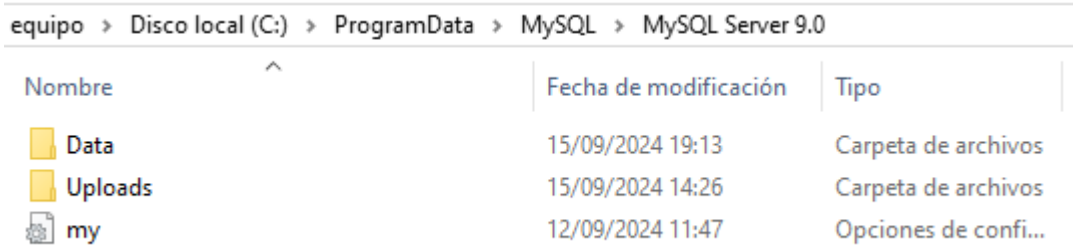
```sql
CREATE TABLE IF NOT EXISTS users_all (
    id INT auto_increment,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(150),
    email VARCHAR(150),
    birth_date VARCHAR(100),
    country VARCHAR(150),
    city VARCHAR(150),
    postal_code VARCHAR(100),
    address VARCHAR(255),
    PRIMARY KEY(ID)
);

#Luego la tabla products
CREATE TABLE IF NOT EXISTS products (
    id INT PRIMARY KEY,
    product_name VARCHAR(100),
    price DECIMAL(10,2),
    colour VARCHAR(100),
    weight FLOAT,
    warehouse_id VARCHAR(255)
);
```

Una vez creadas las tablas, vamos a empezar a cargar los CSV.
Cuando quise cargar los CSV me encontré con una serie de problemas y los fui solucionando uno por uno hasta efectivamente poder cargar todos.

1.

“Secure-file-priv" y “local_infile” estos fueron los primeros errores que me impedían cargar los CSV.
La solución que he encontrado fue desactivar en ambos comandos desde la carpeta de MySql.



En la carpeta ”Uploads” es donde voy a introducir los CSV para cargarlos desde una consulta.

SeY dentro del archivo  que es un archivo de configuración, hice los siguientes cambios:
En la sección de cliente

```
[client]
local_infile=1
# pipe=
```

Que me habilitara la carga de datos locales desde archivos en el cliente utilizando la sentencia LOAD DATA LOCAL INFILE.
Luego en la sección del servidor

```
secure-file-priv="C:/ProgramData/MySQL/MySQL Server 9.0/Uploads"
secure-file-priv=""
```

Que me permitirá, la primera, acceder a archivos desde la carpeta uploads. Y la segunda, desde cualquier ruta del ordenador.

	Variable_name	Value
▶	secure_file_priv	

Una vez solucionado esto, empiezo a cargar el CSV ”transaction”.

```
122     #Aqui la separacion no es por comas, es por punto y coma...
123 •   LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\transactions.csv"
124     INTO TABLE transaction
125     FIELDS TERMINATED BY ';' OPTIONALLY ENCLOSED BY '"'
126     IGNORE 1 LINES;
```

1. A la hora de cargar los datos, importante que en la ruta estén las 2 barras (\\) porque si no, no toma la ruta. Entonces “LOAD DATA INFILE + RUTA“.
2. Se indica en que tabla introducimos los datos. Entonces ”INTO TABLE transaction”.
3. Luego lo más normal es que un CSV este separado por comas, y me di cuenta que en este caso estaba separados por punto y coma(;). Porque la coma, o el punto y coma en este caso, indica cuando termina un campo. Es el signo de separación entre campo y campo. Entonces ”FIELDS TERMINATED BY ’;’. OPTIONALLY ENCLOSED BY ””. Esto significa que, en los datos que estén entre ” ” se puede incluir una coma.
4. Por último ”IGNORE 1 LINES” que indica que no cuente con la primera fila ya que son los nombres identificatorios de los campos.

```
128 •   SELECT *
129     FROM transaction;  -- Comprobamos que haya cargado con exito.
```

Continuamos con la tabla ”company”:

```
131 •   LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\companies.csv"
132     INTO TABLE company
133     FIELDS TERMINATED BY ','
134     IGNORE 1 LINES;
136 •   SELECT *
137     FROM company;  -- Comprobamos que haya cargado con exito.
```

La siguiente es la tabla credit_card:

```
#Cargado correctamente
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\credit_cards.csv"
INTO TABLE credit_card
FIELDS TERMINATED BY ','
IGNORE 1 LINES;
 SELECT *
 FROM credit_card;  -- Comprobamos que haya cargado con exito.
```

En este caso los datos de la tabla products, en el campo 'Price' estaban con el signo $(dólar) y no se puede insertar un signo dólar dentro de un campo DECIMAL.
Lo que hice fue borrar cada signo de $ que había en el campo Price.
Dentro del Visual Studio Code, con CTRL + D, logre efectivamente hacerlo.

```
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\products.csv"
INTO TABLE products
FIELDS TERMINATED BY ','
IGNORE 1 LINES;
 SELECT *
 FROM products;  -- Comprobamos que haya cargado con exito.
```

Y por último, la carga de los CSV de user_ca, user_usa y user_uk. Por alguna razón, al tener en una fila 2 campos con comillas y dentro comas, no funcionaba la carga porque en MySql leía que a partir de esas comas había información para otro campo. Es decir que, leía más información para más campos. Estos 2 campos fueron, “birth_date” que encontramos información tipo “Mar 20, 2000” y el otro campo era “Address” que por ejemplo tenía la dirección separada por comas dentro de comillas. Luego de intentar hacer muchas pruebas encontré que si borraba las comillas y las comas de “Address” funcionaria la carga. Entonces las quite y efectué la carga con éxito.

```
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\users_ca.csv"
INTO TABLE user_ca
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 LINES;

SELECT *
FROM user_ca;  -- Comprobamos que haya cargado con exito.
```

```sql
#Lo mismo con users_uk
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\users_uk.csv"
INTO TABLE user_uk
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
IGNORE 1 LINES;

SELECT *
FROM user_uk; -- Comprobamos que haya cargado con exito.
LOAD DATA INFILE "C:\\ProgramData\\MySQL\\MySQL Server 9.0\\Uploads\\users_usa.csv"
INTO TABLE user_usa
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
IGNORE 1 LINES;

SELECT *
FROM user_usa; -- Comprobamos que haya cargado con exito.
```

Una vez cargado con éxito los CSV, procederé a cargar con datos la tabla que cree que se llama user_all para poder relacionar estas tablas con la tabla transaction.

```sql
-- Insertar los usuarios de United States
INSERT INTO users_all (id, name, surname, phone, email, birth_date, country, city, postal_code, address)
SELECT id, name, surname, phone, email, birth_date, 'United States' AS country, city, postal_code, address
FROM user_usa;
```

Aqui le decimos que inserte en users_all (todos los campos de alguno de los user, en este caso el user_usa) y que country sea 'United States', de la tabla user_usa.
Y hacemos lo mismo con las otras tablas hacia users_all.

```sql
-- Insertar los usuarios de United Kingdom
INSERT INTO users_all (id, name, surname, phone, email, birth_date, country, city, postal_code, address)
SELECT id, name, surname, phone, email, birth_date, 'United Kingdom' AS country, city, postal_code, address
FROM user_uk;
```

```sql
 -- Insertar los usuarios de Canada
 INSERT INTO users_all (id, name, surname, phone, email, birth_date, country, city, postal_code, address)
 SELECT id, name, surname, phone, email, birth_date, 'Canada' AS country, city, postal_code, address
 FROM user_ca;
```

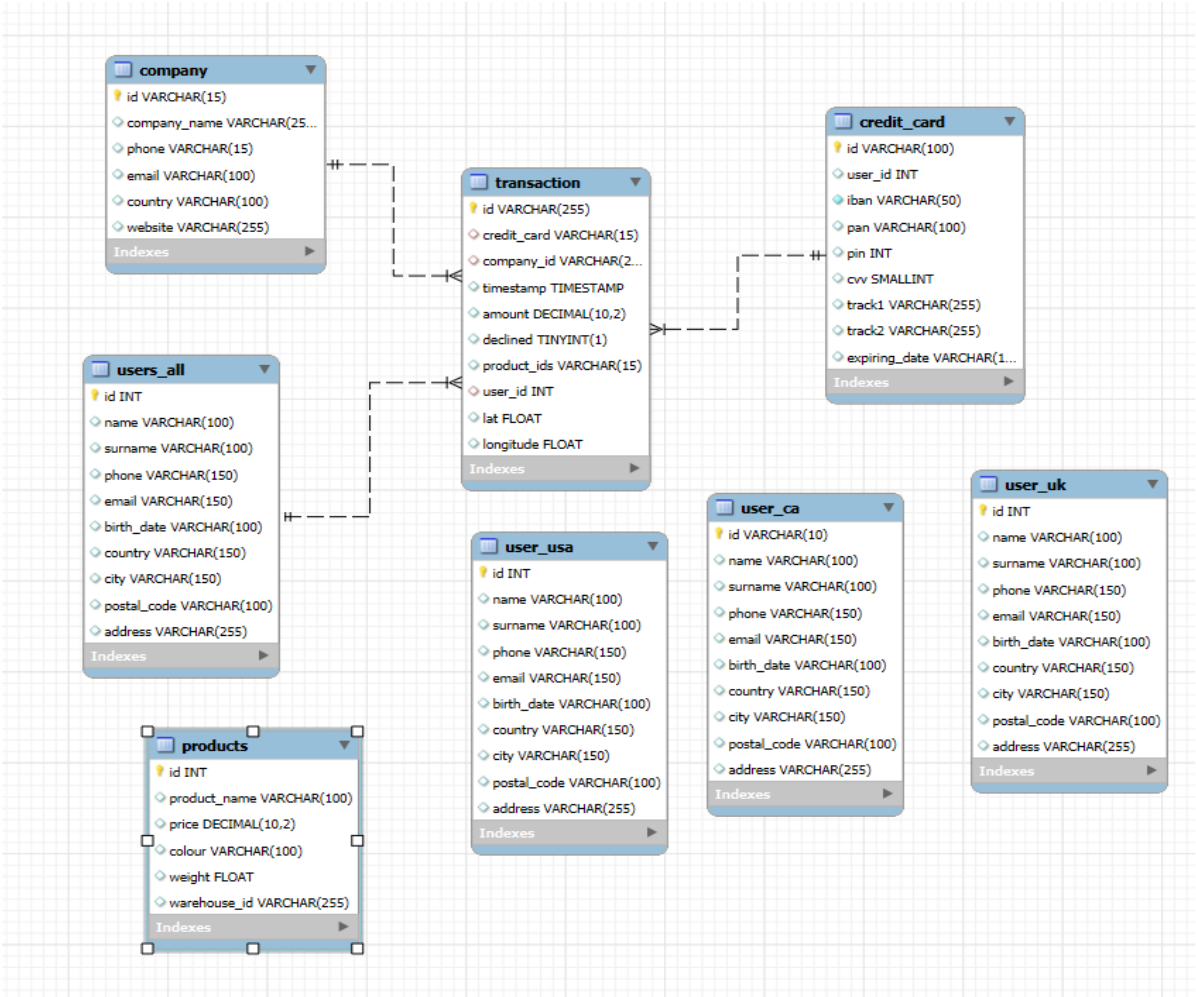
```sql
-- Comprobamos --

SELECT *
FROM users_all
order by id asc;
```

Una vez cargado todos los datos, empezare a relacionar las tablas:

```
219     -- credit_card
220 •   ALTER TABLE transaction
221     ADD CONSTRAINT fk_credit_card
222     FOREIGN KEY (credit_card)
223     REFERENCES credit_card(id);
225     -- company
226 •   ALTER TABLE transaction
227     ADD CONSTRAINT fk_company
228     FOREIGN KEY (company_id)
229     REFERENCES company(id);
231     -- users
232 •   ALTER TABLE transaction
233     ADD CONSTRAINT fk_users_all
234     FOREIGN KEY (user_id)
235     REFERENCES users_all(id);
```

Finalmente, todas las tablas creadas, con sus respectivos sus CSV corregidos para ser efectivamente cargados y tablas relacionadas en un modelo estrella.



- Exercici 1

```sql
-- Ejercicio 1 Realice una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.
SELECT *
FROM users_all
WHERE id IN (
    SELECT user_id
    FROM transaction
    GROUP BY user_id
    HAVING COUNT(company_id) > 30
);
-- Los usuarios que tienen mas de 30 transacciones son con el id 92,267,272,275 --
```

Para el ejercicio 1, empecé a hacer la subconsulta que sería encontrar los usuarios con más de 30 transacciones realizadas. Lo que hice fue hacer una consulta que me agrupé todos los user_id según la cantidad de transacciones que haya por usuario y luego hice un recuento de company_id filtrándolo por los mayores de 30.
Luego una vez que tengo los usuarios que tienen más de 30 transacciones, hago la relación con la tabla de usuarios, para saber quiénes son los usuarios.

El resultado:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
92	Lynn	Riddle	1-387-885-4057	vitae.aliquet@outlook.edu	Sep 21, 1984	United States	Bozeman	61871	P.O. Box 712 7907 Est St.
267	Ocean	Nelson	079-481-2745	aenean@yahoo.com	Dec 26, 1991	Canada	Charlottetown	85X 3P4	Ap #732-8357 Pede Rd.
272	Hedwig	Gilbert	064-204-8788	sem.eget@icloud.edu	Apr 16, 1991	Canada	Tuktoyaktuk	Q4C 3G7	P.O. Box 496 5145 Sapien Ro
275	Kenyon	Hartman	082-871-7248	convallis.ante.lectus@yahoo.com	Aug 3, 1982	Canada	Richmond	R8H 2K2	8564 Facilisi. St.
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Exercici 2

```sql
-- Ejercicio 2 Muestra el promedio del propietario de la tarjeta de crédito IBAN en Donec Ltd, utiliza al menos 2 tablas.
```

Para este ejercicio, entiendo que como resultado final tendré que tener el promedio del amount (avg(amount)) el IBAN del usuario que este en la compañía ”Donec Ltd”.
Entonces pienso que de por si tendré que usar:
Tabla transaction para el avg(Amount).
Tabla credit_card para el IBAN.
Tabla company para el nombre de la compañía.

```sql
SELECT avg(amount), credit_card.iban
FROM transaction
JOIN company
ON transaction.company_id = company.id
JOIN credit_card
ON transaction.credit_card = credit_card.id
JOIN users_all
ON credit_card.user_id = users_all.id
WHERE company.company_name = "Donec ltd"
GROUP BY credit card.iban;
```

Lo que hice fue hacer JOIN con las tablas que necesitaba, luego le dije que sea donde el nombre de la compañía sea “Donec ltd” y lo agrupe por el IBAN de credit_card.
El resultado de la consulta:

avg(amount)	iban
203.715000	PT87806228135092429456346

Pero luego tras la corrección, surgió un dilema que no había tenido en cuenta que estaba haciendo el promedio sobre 2 transacciones y que una de ellas había sido declinada.

```sql
SELECT *
FROM transaction
WHERE company_id = "b-2242";
```

id	credit_card	company_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
52B1839C-D594-EB3D-4A72-730B1C8B08F4	CcU-2973	b-2242	2021-07-31 23:03:21	364.61	1	53, 59, 5, 41	275	-55.8151	-139.586
5B0EEF86-B8A1-EFAA-5EE1-27E7DC8F54A4	CcU-2973	b-2242	2022-01-06 01:44:48	42.82	0	23, 19, 71	275	-64.1136	85.2491

Allí nos encontramos con una transacción declinada entones el promedio no estaba del todo correcto, porque ese dinero nunca llego, al menos eso pareciera.

```sql
SELECT avg(amount), credit_card.iban
FROM transaction
JOIN company
ON transaction.company_id = company.id
JOIN credit_card
ON transaction.credit_card = credit_card.id
JOIN users_all
ON credit_card.user_id = users_all.id
WHERE company.company_name = "Donec ltd" and declined = 0
GROUP BY credit_card.iban;
```

Entonces lo que hice fue agregarle una condición más, que me indique donde declined es = 0, o sea no fue declinada y se ha efectuado la transacción.

El resultado final:

	avg(amount)	iban
▶	42.820000	PT87806228135092429456346