Nivell 1

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules de les quals puguis realitzar les següents consultes:

Primero creamos una nueva base de datos denominada “tienda_s4”:

```
5 •    CREATE DATABASE tienda_s4;
```

```
43  13:14:44  CREATE DATABASE tienda_s4                    1 row(s) affected
```

Después procedemos a crear las tablas y a hacer la importación de los datos.

Para poder realizar la importación de los datos, deberemos comprobar primero en qué carpeta local de nuestro equipo debemos colocar los archivos y verificar si tenemos permisos para la subida de archivos locales:

```
8      -- Comprobamos donde podemos poner los archivos en local para poder subirlos:
9 •   SHOW VARIABLES LIKE "secure_file_priv"; -- encuentra el directorio en el que tienes permitido guardar.
10 •  SHOW VARIABLES LIKE "LOCAL_INFILE"; -- verifica si está ON o OFF.
11 •  SET GLOBAL LOCAL_INFILE = "ON"; -- Se cambia a ON y se verifica con la instrucción anterior.
```

Procedemos a crear las tablas y a realizar la subida de datos:

Como la tabla “transactions” será la tabla de hechos de nuestra base de datos, la crearemos la última para poder establecer las relaciones.

Para la creación de tablas, abriremos cada uno de los archivos y verificaremos como se nos presentan los datos. Nos fijaremos en las columnas que contiene y los nombres de estas para especificarlas en cada tabla creada:

- Creamos la **tabla “companies”**

```
38 • CREATE TABLE IF NOT EXISTS companies (
39    company_id VARCHAR(50) PRIMARY KEY,
40    company_name VARCHAR(250),
41    phone VARCHAR(20),
42    email VARCHAR(100),
43    country VARCHAR(100),
44    website VARCHAR(100)
45    );
```

```
73  11:46:02  CREATE TABLE IF NOT EXISTS companies (    company_id VARCHAR(50) PRI...  0 row(s) affected
```

Subimos los datos:

```
46      -- Subimos los datos de companies
47 •    LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/companies.csv'
48      INTO TABLE companies
49      FIELDS TERMINATED BY ','
50      ENCLOSED BY '"'
51      LINES TERMINATED BY '\n'
52      IGNORE 1 ROWS;
```

```
82  12:00:58  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACAD...  100 row(s) affected Records: 100  Deleted: 0  Skipped: 0  Warnings: 0
```

- Creamos la **tabla "credit_cards":**

```
73 •  CREATE TABLE IF NOT EXISTS credit_cards (
74      id VARCHAR(50) PRIMARY KEY,
75      user_id VARCHAR(20),
76      iban VARCHAR(50),
77      pan VARCHAR(40),
78      pin VARCHAR(4),
79      cvv INT,
80      track1 VARCHAR(255),
81      track2 VARCHAR(255),
82      expiring_date VARCHAR(255)
83      );
```

```
100  12:42:14  CREATE TABLE IF NOT EXISTS credit_cards ( id VARCHAR(50) PRIMARY KEY, u...  0 row(s) affected
```

Subimos los datos:

```
86 •  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/credit_cards.csv'
87    INTO TABLE credit_cards
88    FIELDS TERMINATED BY ','
89    ENCLOSED BY '"'
90    LINES TERMINATED BY '\n'
91    IGNORE 1 ROWS;
```

```
101  12:42:17  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACAD...  5000 row(s) affected Records: 5000  Deleted: 0  Skipped: 0  Warnings: 0
```

- Detectamos que los archivos european_users y american_users contienen las mismas columnas y además no se duplican los id de cada registro, así que unificamos la información y la subimos toda en una misma tabla para una mayor eficiencia. Lo hacemos de la siguiente manera:

Primero creamos la tabla "users", añadiendo una columna extra dónde especificaremos con la subida de información, el continente del usuario (esta información nos lo indica el nombre de cada archivo proporcionado):

```sql
CREATE TABLE IF NOT EXISTS users (
id VARCHAR(50) 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),
continent VARCHAR (50)
);
```

```
109  13:42:41  CREATE TABLE IF NOT EXISTS users ( id VARCHAR(50) PRIMARY KEY, name V...  0 row(s) affected
```

Subimos datos del archivo de usuarios europeos, añadiendo la nueva columna especificando usuarios EUROPE:

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/european_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, name, surname, phone, email, birth_date, country, city, postal_code, address)
SET continent = 'EUROPE';
```

```
110  13:43:14  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACAD...  3990 row(s) affected Records: 3990  Deleted: 0  Skipped: 0  Warnings: 0
```

Subimos datos del archivos de usuarios americanos, añadiendo la nueva columna especificando usuarios AMERICA:

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/american_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, name, surname, phone, email, birth_date, country, city, postal_code, address)
SET continent = 'AMERICA';
```

```
111  13:44:54  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACAD...  1010 row(s) affected Records: 1010  Deleted: 0  Skipped: 0  Warnings: 0
```

- Por último, creamos la tabla “transactions”. Como tenemos el resto de tablas ya creadas, establecemos con la creación de la tabla las Foreign Key y la Primary Key:

```
CREATE TABLE IF NOT EXISTS transactions (
id VARCHAR(100) PRIMARY KEY,
card_id VARCHAR(20),
business_id VARCHAR(255),
timestamp TIMESTAMP,
amount DECIMAL(10,2),
declined TINYINT,
product_ids VARCHAR(255),
user_id VARCHAR(20),
lat FLOAT,
longitude FLOAT,
FOREIGN KEY (card_id) REFERENCES credit_cards(id),
FOREIGN KEY (business_id) REFERENCES companies(company_id),
FOREIGN KEY (user_id) REFERENCES users(id)
);
```

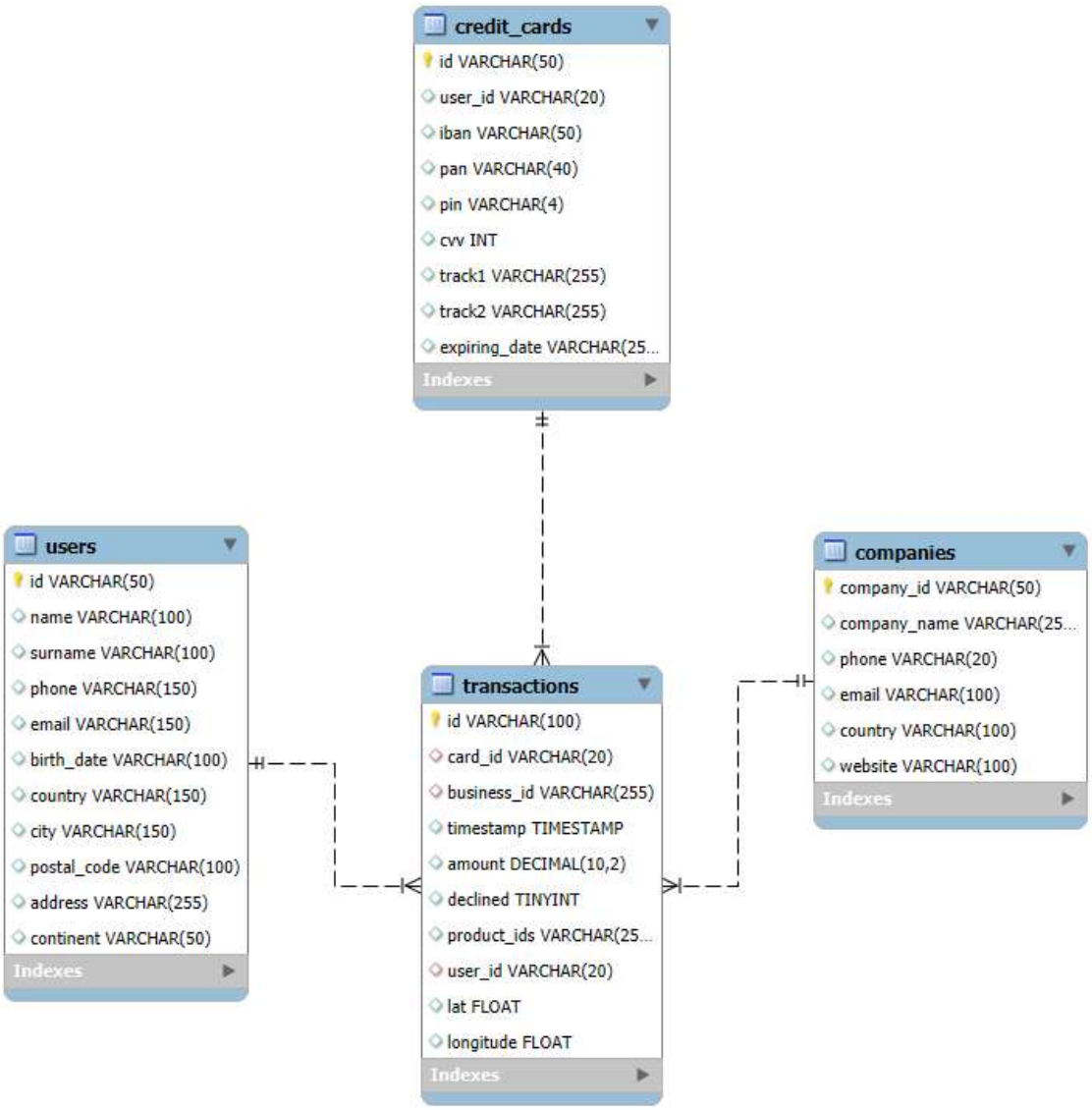
```
115  10:52:03  CREATE TABLE IF NOT EXISTS transactions ( id VARCHAR(100) PRIMARY KEY, ...  0 row(s) affected
```

Ahora procedemos a subir los datos. Como el archivo csv está en formato columnas, en el comando especificamos a qué columna corresponde cada campo y además el separador lo hacemos con ‘;’. Como en el resto de tablas, también especificamos que queremos ignorar la primera fila ya que esta indica solamente el nombre de cada columna pero no es información que queramos incorporar en los datos:

```
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/transactions.csv"
INTO TABLE transactions
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, card_id, business_id, timestamp, amount, declined, product_ids, user_id, lat, longitude);
```

```
117  10:53:20  LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACA...  100000 row(s) affected Records: 100000  Deleted: 0  Skipped: 0  Warnings: 0
```

Finalmente obtenemos el siguiente esquema en estrella en nuestra nueva base de datos:



Vemos que se trata de un esquema en forma de estrella: una tabla de hechos (la tabla “transactions”) y tres tablas de dimensiones que se relacionan con ella.

Las tres tablas de dimensiones tienen una relación con “transactions” de 1:n y en cuanto a las claves primarias y las claves foráneas, se estructura de la siguiente manera:

Tabla “users” – id – PK → Tabla “transactions” – user_id – FK

Tabla “credit_cards” – id – PK → Tabla “transactions” – card_id – FK

Tabla “companies ” – company_id – PK → Tabla “transactions” – “business_id”

Tabla “transactions” – id - PK

Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.

Utilizamos la tabla “users” para mostrar el nombre y apellido de los usuarios y a través de una subquery, realizaremos el filtro de más de 80 transacciones, que se encuentra en la tabla “transactions”:

```
119 •   SELECT name, surname
120     FROM users
121     WHERE users.id IN (SELECT user_id
122                        FROM transactions
123                        GROUP BY user_id
124                        HAVING COUNT(transactions.id) > 80);
```

name	surname
Molly	Gilliam
Dxwgi	Hwcru
Bnyr	Astuw
Sfzzoh	Xgvfridxs

```
150  11:58:33  SELECT name, surname FROM users WHERE users.id IN (SELECT user_id  FROM ...  4 row(s) returned
```

Exercici 2

Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.

Realizamos la media del total de amount y mostramos el iban y el nombre de la empresa que se encuentran en la tabla “credit_cards”, en la tabla “transactions” y en la tabla “companies”. Filtramos que únicamente queremos mostrar la compañía Donec LTD y mostramos la información agrupando por iban.

```
129 •   SELECT ROUND(AVG(amount), 2) AS AvgAmount, iban, company_name
130     FROM credit_cards
131     JOIN transactions ON credit_cards.id = card_id
132     JOIN companies ON company_id = business_id
133     WHERE company_name = 'Donec Ltd'
134     GROUP BY iban;
```

AvgAmount	iban	company_name
356.25	XX9114064011255863075868O5	Donec Ltd
142.96	SK9446370242474562577506	Donec Ltd
257.37	XX7767529178459529755556 40	Donec Ltd
139.59	XX4138273622897193049089 90	Donec Ltd
240.41	XX3477872460707696107803 08	Donec Ltd
188.58	XX6887684365430908948546 02	Donec Ltd

```
20  23:28:23  SELECT ROUND(AVG(amount), 2) AS AvgAmoun...  371 row(s) returned
```

Nivell 2

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades i genera la següent consulta:

Crearemos una nueva tabla dónde se especifique si la tarjeta está activada o desactivada. Esta tabla la creamos a través de una consulta con datos de tablas ya creadas.

Primero buscamos una consulta que nos ayude a obtener una columna nueva donde se determina si la tarjeta está activada o no en base a los datos aportados en el ejercicio.

Mostramos la información de card_id y obtenemos una nueva columna que especifica si la tarjeta está activada o no a través del CASE: si el declined da una suma total de tres, entonces ya se considera una tarjeta desactivada, si no se cumple será una tarjeta Activada.

Para poder analizar cada tarjeta, se hace una iteración a través del ROW_NUMBER, que irá analizando cada card_id, ordenado la transacción de más actual a más antiguo y filtramos para que sólo nos de las tres últimas en el tiempo:

```
142 •   SELECT card_id,
143     CASE
144         WHEN SUM(declined) = 3 THEN 'Inactivated'
145         ELSE 'Activated'
146     END AS status_card
147     FROM
148         (SELECT card_id, declined, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_card
149         FROM transactions
150         ) AS transactions_date
151     WHERE transactions_date.row_card <= 3
152     GROUP BY card_id;
```

	card_id	status_card
	CcS-4870	Inactivated
	CcS-4899	Inactivated
	CcS-4998	Inactivated
	CcS-5035	Inactivated
	CcU-3568	Inactivated
	CcS-4857	Activated

15 13:03:28 SELECT card_id, CASE WHEN SUM(declined) = 3 THEN 'Inactivated' ELSE 'Acti... 5000 row(s) returned

Con esta consulta, procedemos a crear la nueva tabla “status_cards”:

```
156 •  CREATE TABLE status_cards AS
157    SELECT card_id,
158    CASE
159        WHEN SUM(declined) = 3 THEN 'Inactivated'
160        ELSE 'Activated'
161    END AS status_card
162    FROM
163        (SELECT card_id, declined, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_card
164        FROM transactions
165        ) AS transactions_date
166    WHERE transactions_date.row_card <= 3
167    GROUP BY card_id;
```

17 13:06:17 CREATE TABLE status_cards AS SELECT card_id, CASE WHEN SUM(declined) = ... 5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

Comprobamos que se nos ha creado la nueva tabla:


```
170 •  SHOW COLUMNS FROM
171    status_cards;
```

Field	Type	Null	Key	Default	Extra
card_id	varchar(20)	YES		NULL	
status_card	varchar(11)	NO			

18 13:07:39 SHOW COLUMNS FROM status_cards 2 row(s) returned

```
173 •  SELECT *
174    FROM status_cards;
```

card_id	status_card
CcS-4857	Activated
CcS-4858	Activated
CcS-4859	Activated
CcS-4860	Activated
CcS-4861	Activated

19 13:08:18 SELECT * FROM status_cards 5000 row(s) returned

Establecemos la relación de la tabla:

Asignamos la PK en la tabla estado_tarjetas

```
178 •   ALTER TABLE status_cards
179     ADD PRIMARY KEY (card_id);
```

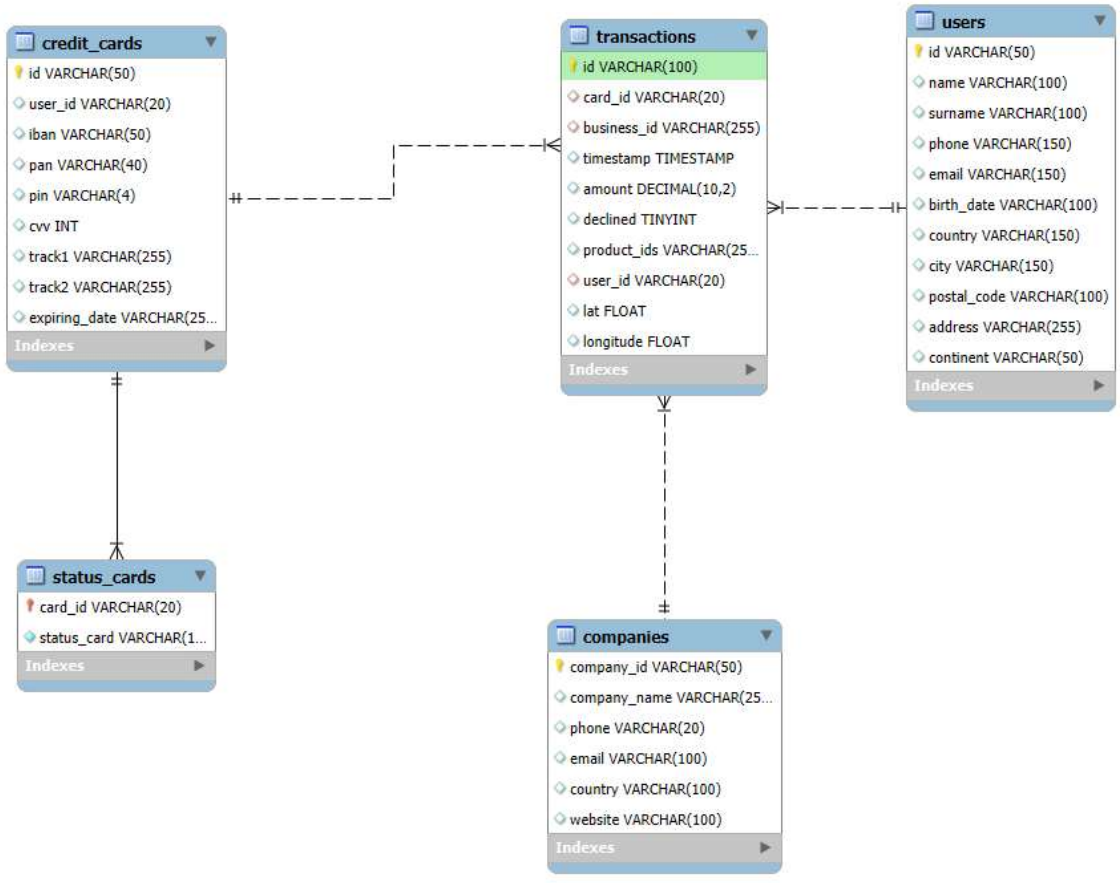
```
32  13:18:59  ALTER TABLE status_cards ADD PRIMARY KEY (card_id)   0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

Asignamos la FK en la tabla status_cards en la misma columna card_id (que también es PK). De esta manera se garantiza que no puede existir un registro en status_cards sin una tarjeta que no se haya creado en credit_cards. Por lo tanto establecemos una relación de 1 a 1.

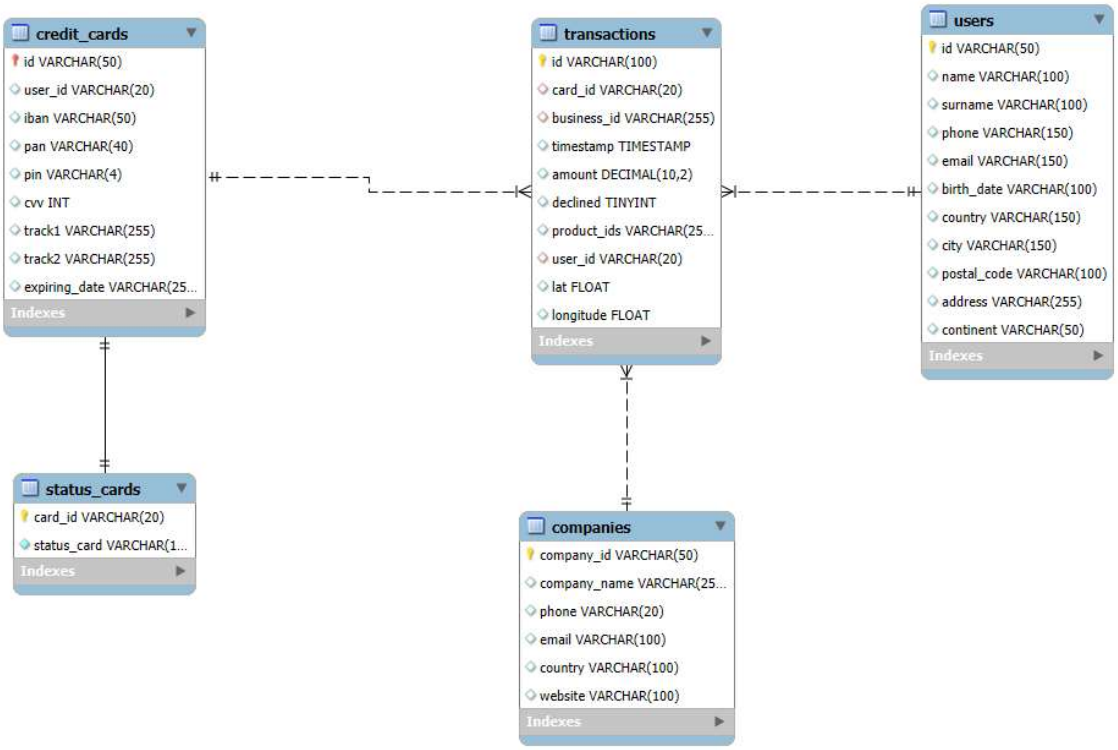
```
182 •   ALTER TABLE status_cards
183     ADD CONSTRAINT fk_status_cards_a_credit_cards
184     FOREIGN KEY (card_id)
185     REFERENCES credit_cards(id);
```

```
11  22:40:48  ALTER TABLE status_cards ADD CONSTRAINT ...  5000 row(s) affected Records: 5000  Duplicates: 0  Warnings: 0
```

Obtenemos el siguiente diagrama:



Establecemos la relación de 1 a 1 de forma manual y el diagrama queda de la siguiente manera:



Ahora obtenemos un diagrama que pasa a ser un copo de nieve.

Como novedad tenemos la nueva tabla creada “status_cards”, que se relaciona con la tabla credit_cards de 1:1.

La PK de credit_cards es id que es la misma FK en status_cards.

Exercici 1

Quantes targetes estan actives?

Contamos el número de tarjetas de la tabla “status_cards” filtrando únicamente las Activadas.

```
190 •   SELECT COUNT(status_card)
191     FROM status_cards
192     WHERE status_card = 'Activated';
```

	COUNT(status_card)
▶	4995

37 13:25:54 SELECT COUNT(status_card) FROM status_cards WHERE status_card = 'Activate... 1 row(s) returned

Nivell 3

Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

Primero procedemos a crear y a hacer la subida de información de la tabla “products”.

Creamos la tabla:

```
203 • CREATE TABLE IF NOT EXISTS products (
204     id VARCHAR(50) PRIMARY KEY,
205     product_name VARCHAR(250),
206     price VARCHAR(20),
207     colour VARCHAR(100),
208     weight DECIMAL(10,2),
209     warehouse_id VARCHAR(100)
210 );
```

```
85  12:14:58  CREATE TABLE IF NOT EXISTS products (    id VARCHAR(50) PRIMARY KEY, ...  0 row(s) affected
```

Subimos datos:

```
213 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACADEMY/products.csv'
214   INTO TABLE products
215   FIELDS TERMINATED BY ','
216   ENCLOSED BY '"'
217   LINES TERMINATED BY '\n'
218   IGNORE 1 ROWS;
```

```
86  12:16:16  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ITACAD...  100 row(s) affected Records: 100  Deleted: 0  Skipped: 0  Warnings: 0
```

Cuando vamos a establecer la relación entre la tabla “products” y la tabla de hechos “transactions”, detectamos que en el archivo, en la columna de “producto_ids”, los ids de productos de cada transacción se encuentran guardados en la misma celda y separados por comas.

Para poder resolver esto, crearemos una tabla puente en la que se establecerá una relación de n a n y que irá desde la tabla “transactions” hasta la tabla “products.

Primero creamos la tabla puente, denominada “transaction_products” y establecemos la relación FK con las tablas “transactions” y “products”:

```
218 • CREATE TABLE transaction_products(
219     transaction_id VARCHAR(255),
220     product_id VARCHAR(50),
221     FOREIGN KEY (transaction_id) REFERENCES transactions(id),
222     FOREIGN KEY (product_id) REFERENCES products(id)
223     );
```

```
40  10:34:45  CREATE TABLE transaction_products( transaction_id VARCHAR(255), product_id V...  0 row(s) affected
```

Asignamos las PK de la tabla creada:

```
225 •   ALTER TABLE transaction_products
226     ADD PRIMARY KEY (transaction_id, product_id);
```

```
42  11:08:41  ALTER TABLE transaction_products ADD PRIMARY KEY (transaction_id, product_id)  0 row(s) affected Records: 0  Duplicates: 0  Warnings: 0
```

A continuación, procedemos a realizar la carga de datos en esta nueva tabla.

Los datos los extraeremos de la columna id y de la columna product_ids de la tabla “transactions”.

Especificamos que queremos añadir los datos de id de transacción y de id de producto de la tabla “transactions” a la nueva tabla creada “transaction_products” a la columna transaction_id y a la columna producto_id

Utilizaremos la sentencia FIN IN SET para poder recoger los datos ya subidos con anterioridad.

Para una correcta ejecución y que el comando interprete correctamente, es necesario que en la columna product_ids no exista ningún espacio entre cada carácter ya que los espacios actúan como delimitadores para separar argumentos, por ello especificamos los delimitadores.

```
233 •   INSERT IGNORE INTO transaction_products(transaction_id, product_id)
234     SELECT transactions.id, products.id
235     FROM transactions
236     JOIN products ON FIND_IN_SET(products.id, replace(product_ids,' ',' '));
```

```
8  15:25:30  INSERT IGNORE INTO transaction_products(tran...  253391 row(s) affected Records: 253391  Duplicates: 0  Warnings: 0
```

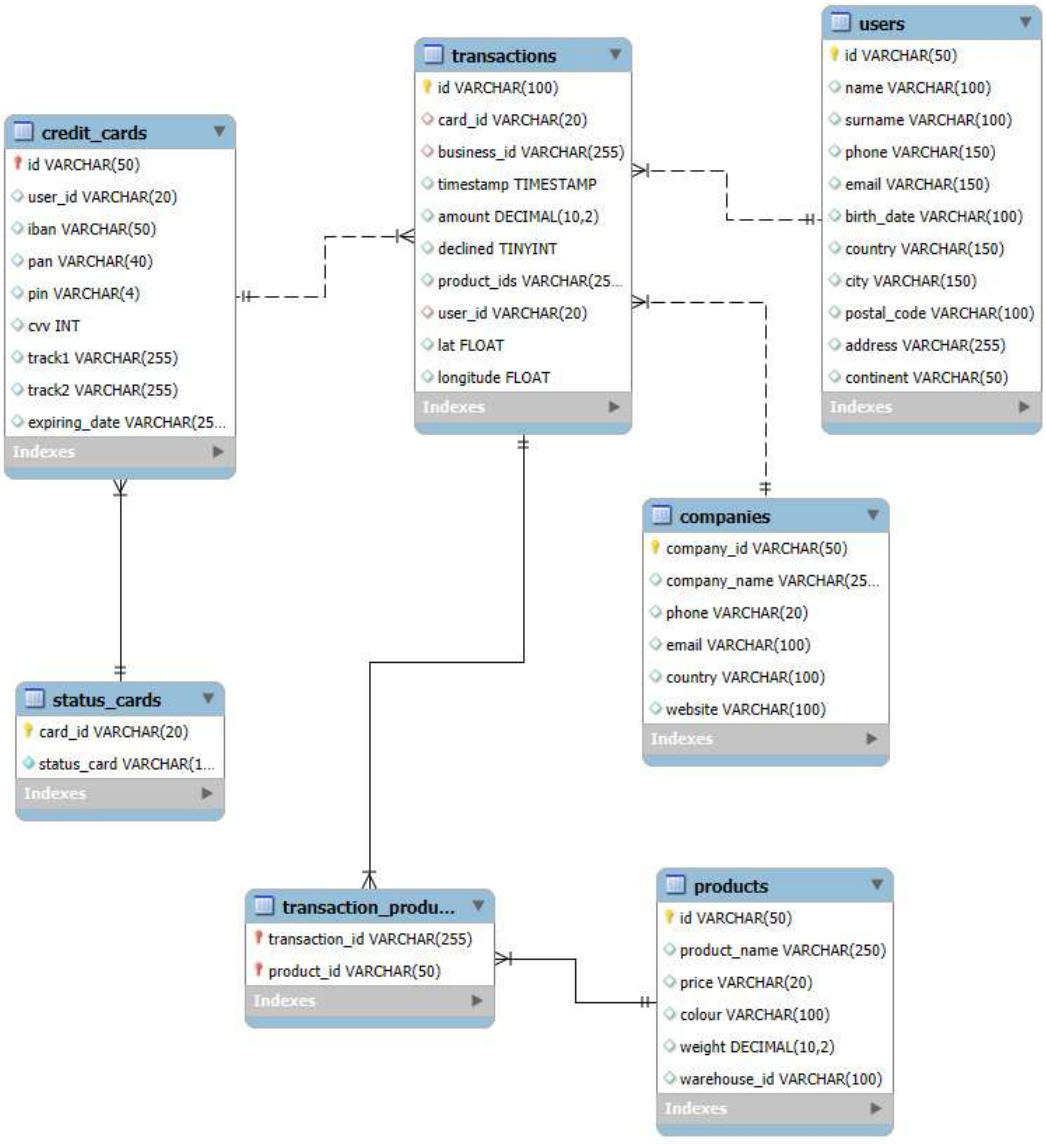
Comprobamos que los datos se han subido correctamente a la nueva tabla:

```
243 •   SELECT *
244     FROM transaction_products;
```

transaction_id	product_id
001A60EA-DC9C-4E5A-9460-6628B100E7E1	1
0032F0BB-BBE6-4AA5-B5EE-EEAD533C0C48	1
00342381-503D-422D-85AB-F2D4FFAAD4C7	1
004C0A80-E537-46D8-BE44-343D2176DF15	1
004D1DB5-B2CB-4460-98B6-31C42CA96E5F	1
0062599C-0A1F-4405-AD55-45E20043B551	1
007B1297-C3ED-4966-802C-68FE106AD25C	1
007EA15D-AA1F-4FB3-9483-7D055E99CEC0	1

58 11:51:46 SELECT * FROM transaction_products 253391 row(s) returned

Una vez introducida la tabla “products” y la tabla “transaction_products”, el diagrama de la base de datos queda de la siguiente manera:



Vemos que hemos añadido una nueva tabla a nuestro diagrama en forma de copo de nieve.

Como se puede observar, esta nueva tabla “transaction_products” es una tabla puente que une “transactions” con “products”, permitiendo establecer una relación de n:n.

En la tabla “transaction_products”, las dos columnas que presenta son PK. La FK de transaction_id se encuentra en el id de la tabla “transactions” y la FK de “product_id” se encuentra en el id de la tabla “products”, conectando así una tabla con la otra.

Exercici 1

Necessitem conèixer el nombre de vegades que s'ha venut cada producte.

Realizamos JOIN de tres tablas para poder presentar el nombre del producto, el número total de ventas (a través del COUNT) y filtrar por declined = 0 (que se considera que son las ventas finales).

```
243 •   SELECT product_name, COUNT(transaction_id) AS Sales
244     FROM transaction_products
245     JOIN products ON product_id = products.id
246     JOIN transactions ON transactions.id = transaction_id
247     WHERE declined = 0
248     GROUP BY product_id;
```

product_name	Sales
the duel warden	2602
Stark Karstark	2532
sith Jade	2591
jinn Winterfell	2520
mustafar jinn	2590
Littlefinger the giantsblood	2453