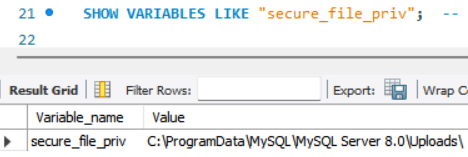
Sprint 4: Modelado SQL

Radostin Pavlov

# NIVEL 1  
  
Descarga los archivos CSV, estudialos y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas de las que puedas realizar las siguientes consultas...

Empezamos creando las tablas e importando los datos desde los archivos csv. Para poder importar los datos de los archivos, hay que comprobar en qué carpeta hay que colocarlos para poder leerlos:



Después se crea la base de datos y todas las tablas. Las longitudes de los campos se ajustan aproximadamente a las de los datos, dejando cierto margen donde procede para nuevos datos. Se asegura que los tipos de las claves principales sean iguales que los de las claves foráneas para poder unir las tablas después de crearlas. Las claves foráneas se especifican tras crear las tablas para evitar errores por no poder establecer las referencias si las tablas correspondientes aun no se han creado.

create database ventas;  
  
use ventas;  
  
CREATE TABLE companies (  
 company\_id VARCHAR(10) PRIMARY KEY,  
 company\_name VARCHAR(40),  
 phone VARCHAR(20),  
 email VARCHAR(50),  
 country VARCHAR(60),  
 website VARCHAR(50)  
);   
  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'  
INTO TABLE companies  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
  
CREATE TABLE credit\_cards (  
 id VARCHAR(10) PRIMARY KEY,  
 user\_id INT,  
 iban VARCHAR(40),  
 pan VARCHAR(30),  
 pin VARCHAR(10),  
 cvv VARCHAR(5),  
 track1 VARCHAR(50),  
 track2 VARCHAR(50),  
 expiring\_date VARCHAR(10)   
);   
  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit\_cards.csv'  
INTO TABLE credit\_cards  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
  
CREATE TABLE products (  
 id INT PRIMARY KEY,  
 product\_name VARCHAR(40),  
 price VARCHAR(10),  
 colour VARCHAR(10),  
 weight VARCHAR(5),  
 warehouse\_id VARCHAR(10)  
);   
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'  
INTO TABLE products  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
  
CREATE TABLE transactions (  
 id VARCHAR(45) PRIMARY KEY,  
 card\_id VARCHAR(10),  
 business\_id VARCHAR(10),  
 timestamp TIMESTAMP,  
 amount DECIMAL(10, 2),  
 declined TINYINT,  
 product\_ids VARCHAR(15),  
 user\_id INT,  
 lat DECIMAL(15, 11),  
 longitude DECIMAL(15, 11)  
);   
  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'  
INTO TABLE transactions  
FIELDS TERMINATED BY ';'  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
  
CREATE TABLE users (  
 id INT PRIMARY KEY,  
 name TINYTEXT,  
 surname TINYTEXT,  
 phone VARCHAR(20),  
 email VARCHAR(40),  
 birth\_date VARCHAR(15),  
 country TINYTEXT,  
 city TINYTEXT,  
 postal\_code VARCHAR(11),  
 address VARCHAR(40)  
);

Al tratar de importar los datos correspondientes a la tabla users mediante código SQL se encuentran errores, al parecer porque mySQL no lee bien los separadores. Para solventarlo, las comas usadas como separadores se sustituyen por semicolons, empleando el siguiente código de Python:

import pandas as pd

input\_file = 'users\_usa.csv'

output\_file = 'users\_usa-2.csv'

df = pd.read\_csv(input\_file, sep=',', quotechar='"')

df.to\_csv(output\_file, sep=';', index=False)

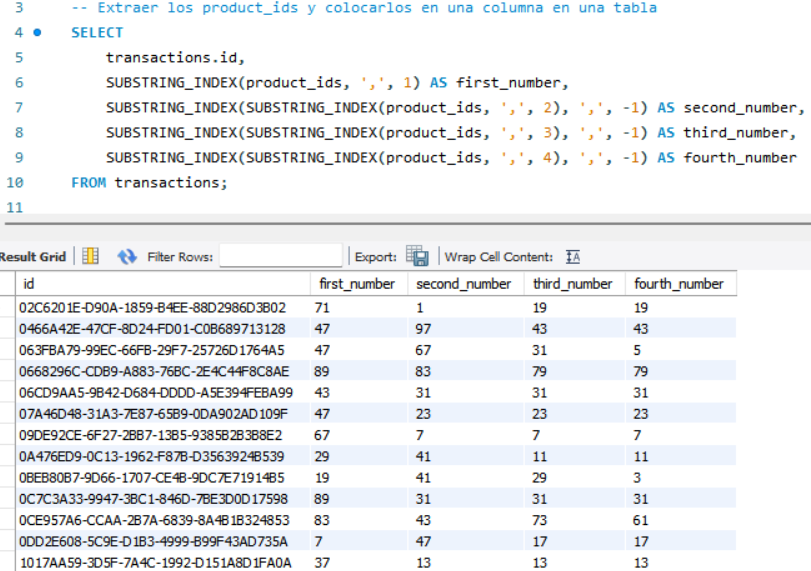
Esto se hace con cada uno de los tres archivos que van a la tabla ususarios. En los nombres de los archivos generados se incluye la extensión -2 para distinguirlos de los originales. A continuación se insertan los datos de los tres archivos en la misma tabla users ya que lo que varía básicamente es el país que aparece en el campo ‘country’.

Corrección: Se encuentra una solución alternativa, que consiste en importar los datos mediante la función Table Data Import Wizard de MySQLWorkbench, creando una tabla nueva en el proceso.

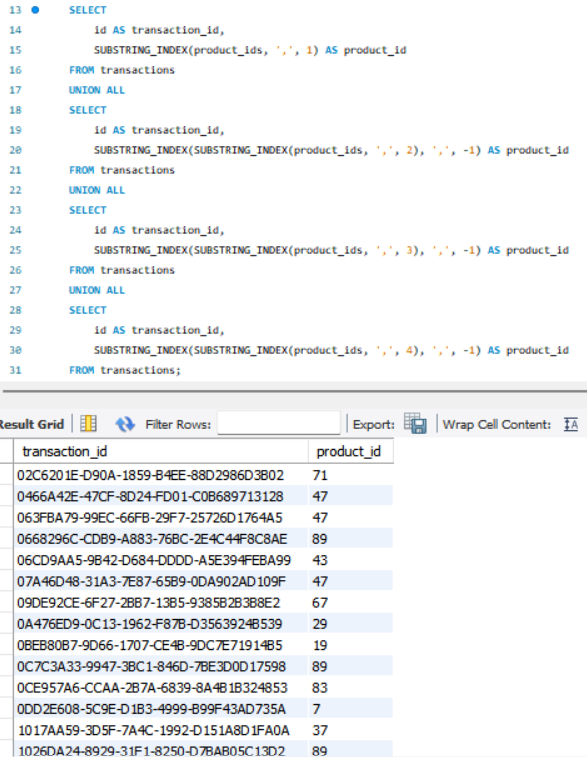
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users\_ca-2.csv'   
  
INTO TABLE users  
FIELDS TERMINATED BY ';'  
-- ENCLOSED BY '"'   
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users\_uk-2.csv'   
-- El archivo original daba problemas, de modo que fue transformado con python   
-- sustituyendo el delimitador coma por semicolon y eliminando comillas  
INTO TABLE users  
FIELDS TERMINATED BY ';'  
-- ENCLOSED BY '"'   
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users\_usa-2.csv'   
-- El archivo original daba problemas, de modo que fue transformado con python   
-- sustituyendo el delimitador coma por semicolon y eliminando comillas  
INTO TABLE users  
FIELDS TERMINATED BY ';'  
-- ENCLOSED BY '"'   
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;  
  
  
A continuación se genera una tabla llamada products\_transactions conteniendo las claves transaction\_id y product\_id de las tablas products y transactions para romper la relación n:n entre ellas. (Esto corresponde al Nivel 3, pero lo hacemos aquí y así ya completamos el modelo). La tabla se genera mediante Python o SQL. El código de Python usado se puede ver en el archivo ‘id-s-to-table.ipynb’ incluido en el repositorio. Básicamente consiste en crear con str.split() una tabla con una columna por cada valor de la lista del campo product\_ids, insertar los transaction\_id en la tabla con insert() y pivotar los datos de los product id-s colocándolos en una sola columna con melt(). El dataframe se exporta como ‘products\_transactions.csv’ que luego se importa en la tabla products\_transactions creada.  
  
  
CREATE TABLE products\_transactions (  
 transaction\_id VARCHAR(45),  
 product\_id INT  
);   
  
  
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products\_transactions.csv'  
INTO TABLE products\_transactions  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
IGNORE 1 ROWS;

La alternativa por sql se ha realizado de la siguiente manera:

Primero se ha extraído cada uno de los números del campo transactions.product\_ids junto al transactions.id correspondiente:



A continuación, se modifica el código para colocar los diferentes substrings en una sola columna empleando union all (esto sería el eqivalente a melt() in python).



Para convertir esta consulta en una tabla dentro de la base de datos, se añade un CREATE TABLE al principio de la consulta:



A continuación se crean las claves foráneas:

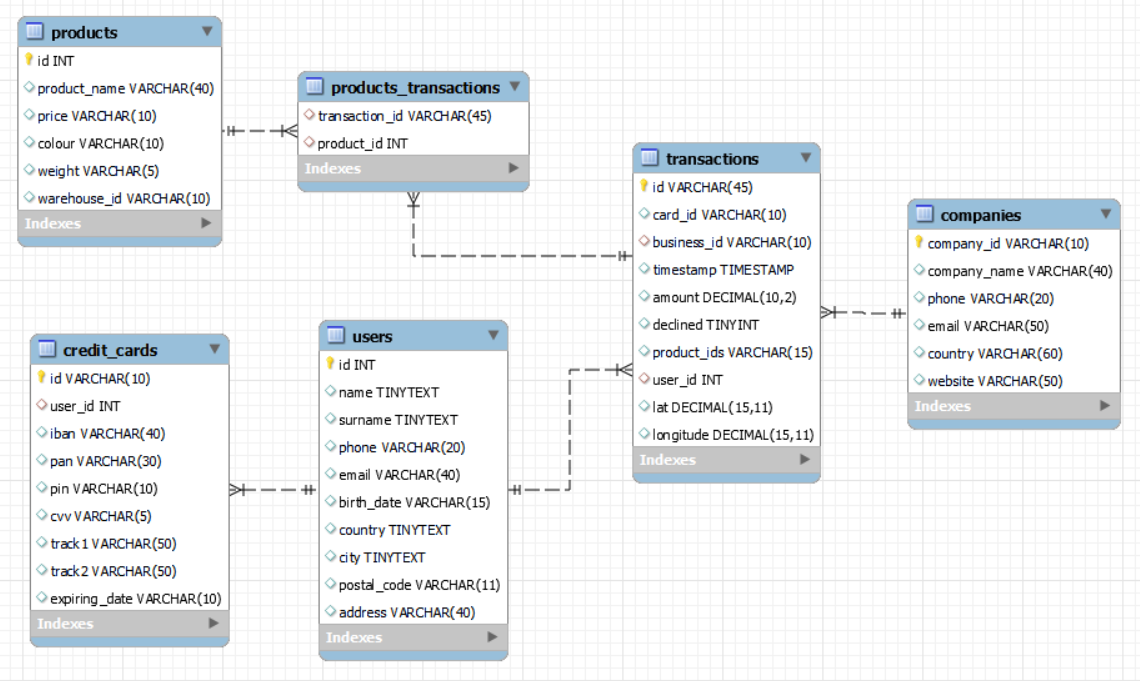
ALTER TABLE credit\_cards ADD FOREIGN KEY(user\_id) REFERENCES users(id);

Corrección: la tabla credit\_cards ha de ir vinculada directamente a transactions. Por lo tanto, se borra el vínculo credit\_cards.user\_id con users.id y se crea uno nuevo entre entre transactions.card\_id y credit\_cards.id:

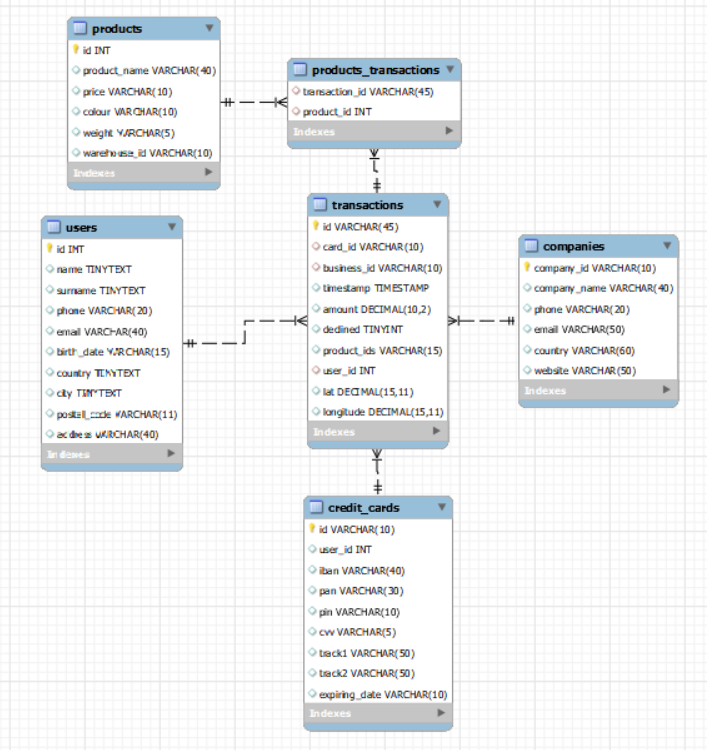
ALTER TABLE transactions ADD FOREIGN KEY(card\_id) REFERENCES credit\_cards(id);

ALTER TABLE transactions ADD FOREIGN KEY (user\_id) REFERENCES users(id);  
ALTER TABLE transactions ADD FOREIGN KEY (business\_id) REFERENCES companies(company\_id);  
ALTER TABLE products\_transactions ADD FOREIGN KEY (transaction\_id) REFERENCES transactions(id);  
ALTER TABLE products\_transactions ADD FOREIGN KEY (product\_id) REFERENCES products(id);

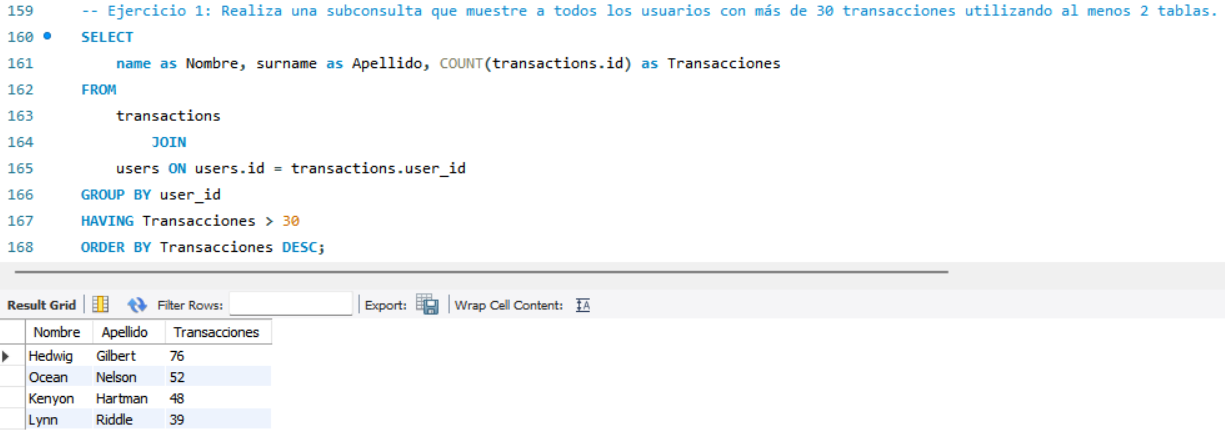
Una vez creadas las tablas y sus relaciones se genera el modelo con la función Reverse Engineering de MySQL Woorkbench:

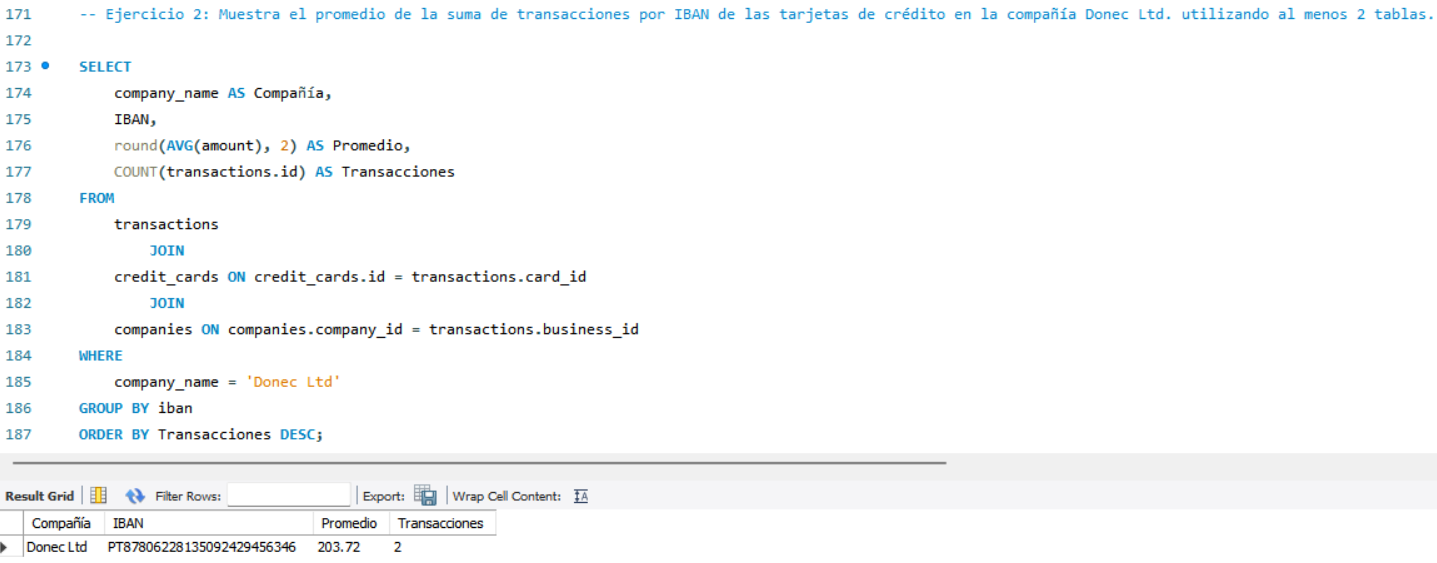


Modelo corregido:



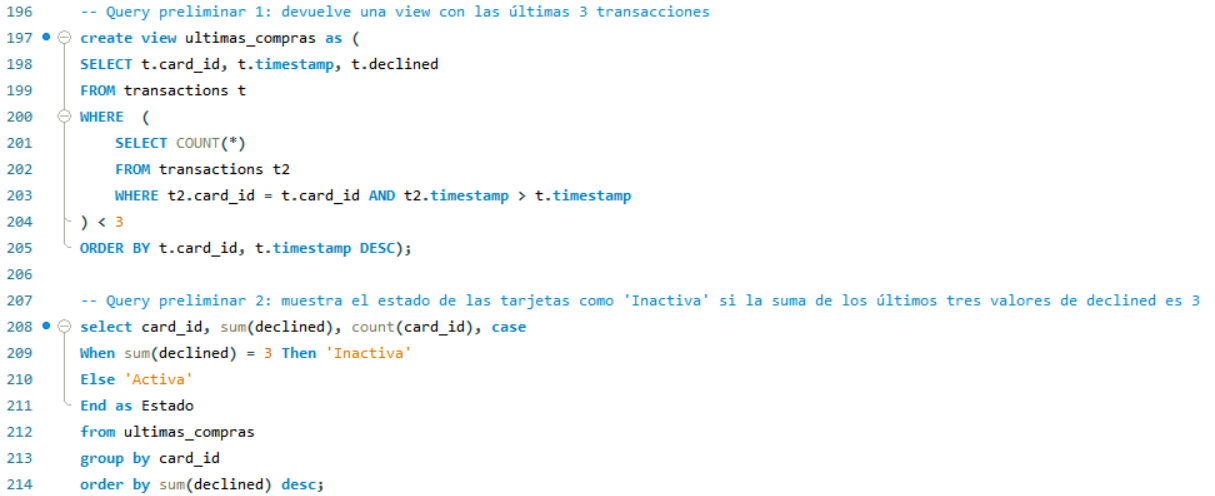
Se comprueba que todas las relaciones son correctas y se procede a realizar las consultas solicitadas.

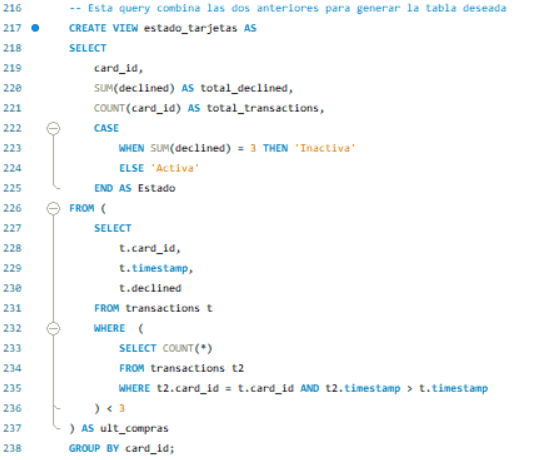




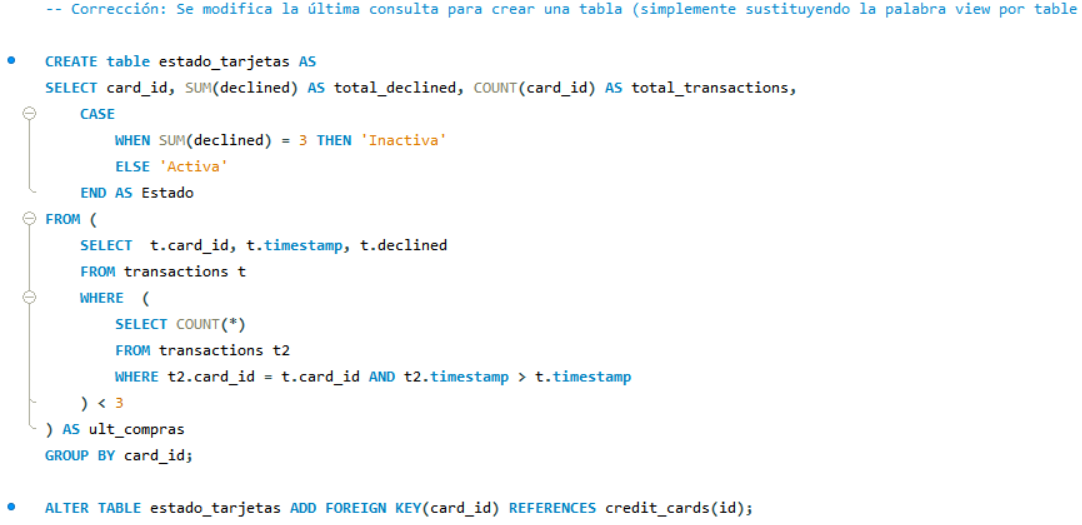
NIVEL 2  
-- Crea una tabla nueva que refleje el estado de las tarjetas de crédito en base a si   
-- las últimas tres transacciones fueron declinadas y genera la siguiente consulta:  
-- Ejercicio 1: Cuantas tarjetas están activas

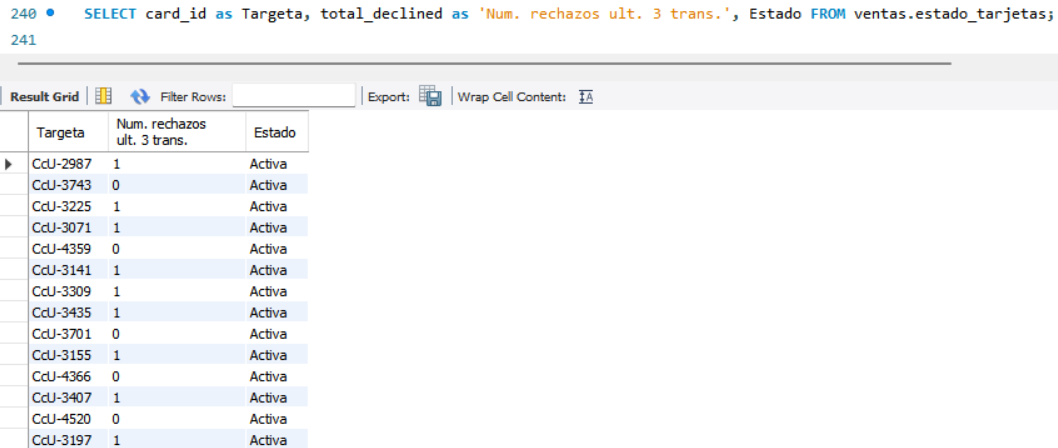
Para resolver este ejercicio se han creado dos queries preliminares que luego se han combinado en una sola:

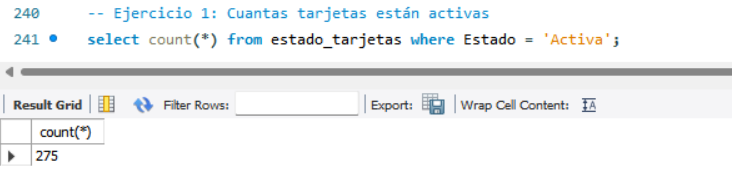




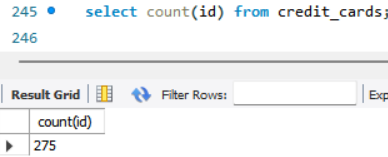
Corrección: la última consulta se modifica para crear una tabla en vez de una vista (simplemente sustituyendo la palabra view por table). Además, se puede vincular la tabla creada estado\_tarjetas a la tabla credit\_cards o a la tabla transactions:

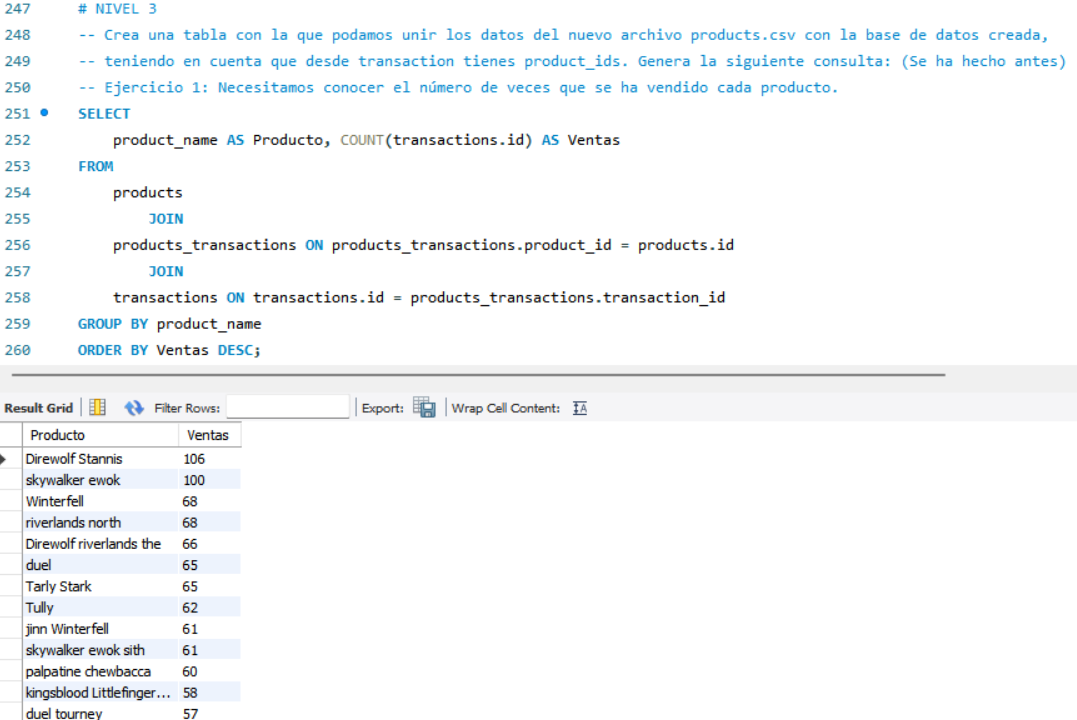






Se comprueba que este número corresponde a la totalidad de las tarjetas registradas:





+