DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

SPRINT 04

Creación de base de datos
12 de mayo de 2025

Letiane Benincá
benincalf@gmail.com

IT ACADEMY
Barcelona Activa

TAREA S4.01
CREACIÓN DE BASE DE DATOS
DESCRIPCIÓN

Partiendo de algunos archivos CSV diseñaras y crearas la tuya base de datos.

NIVEL 1

Descarga los archivos CSV, estúdialos y diseña una base de datos con un esquema de estrella que contenga, como mínimo 4 tablas de las cuales puedes realizar las siguientes consultas:

Inicialmente revisamos los datos que tenemos y revisamos los campos de las tablas:

TABLAS				
transaction	**companies**	**credit_cards**	**user (ca, uk, usa)**	**products [NIVEL 3]**
id	*company_id*	*id*	*id*	*id*
card_id	company_name	*user_id*	name	product_name
business_id	phone	iban	surname	price
timestamp	email	pan	phone	colour
amount	country	pin	email	weight
declined	website	cvv	birth_date	warehouse_id
product_ids		track1	country	
user_id		track2	city	
lat		expiring_date	postal_code	
longitude			address	

Tenemos los campos de cada tabla, sus primary keys y foreign keys (en amarillo), para relacionarse entre ellas. Se identifica que los datos de los users son iguales, así que unificamos en solo una tabla. La tabla products, será incorporada para los ejercicios del nivel 3.

Para empezar a crear el database, tenemos que entender como son los datos que serán inseridos, para utilizar la codificación correcta:

** Recomendable inicializar los campos alfanuméricos con VARCHAR(255), para evitar errores en la carga y luego adaptarlos a los tipos de datos que vas a utilizar.*

TABLA COMPANY				
COLUMNA	**DATOS A INSERIR**	**TIPOLOGIA**	**USADO**	**RELACCIÓN**
company_id	b-2222	Alfanumérico	VARCHAR(255)	**PRIMARY KEY**
company_name	Ac Fermentum Incorporated	Alfanumérico	VARCHAR(255)	
phone	06 85 56 52 33	Alfanumérico	VARCHAR(255)	
email	donec.porttitor.tellus@yahoo.net	Alfanumérico	VARCHAR(255)	
country	Germany	Alfanumérico	VARCHAR(255)	
website	https://instagram.com/site	Alfanumérico	VARCHAR(255)	

TABLA CREDIT CARDS				
COLUMNA	**DATOS A INSERIR**	**TIPOLOGIA**	**USADO**	**RELACCIÓN**
id	CcU-2945	8 caracteres alfanuméricos.	VARCHAR(255)	**PRIMARY KEY**
user_id	274	Entero	INT	**FOREING KEY**
iban	DO26854763748537475216568689	Alfanumérico	VARCHAR(255)	
pan	5,14242E+15	Alfanumérico	VARCHAR(255)	
pin	9080	*Alfanumérico*	VARCHAR(255)	
cvv	887	*Alfanumérico*	VARCHAR(255)	
track1	%B4621311609958661^UftuyfsSeimxn ^061062824l?7	Alfanumérico	VARCHAR(255)	
track2	%B4149568437843501=5107140330?1	Alfanumérico	VARCHAR(255)	
expiring_date	08/24/23	Alfanumérico, en este caso no está en patrón de DATE.	VARCHAR(255)	

* *En la peer review, comentamos que pin y pan, podrían ser VARCHAR, cambiando de INT, porque si en algún momento hay un número que se inserte con el 0, a principio, no lo tendría en cuenta. Siendo VARCHAR, si mantiene el número (ejemplo: 012).*

Para la tabla user, unificamos la información que es igual en los tres excels:

TABLA USERS (ca, uk, usa)				
COLUMNA	**DATOS A INSERIR**	**TIPOLOGIA**	**USADO**	**RELACCIÓN**
id	201	Entero	INT AUTO_INCREMENT	**PRIMARY KEY**
name	Iola	Alfanumérico	VARCHAR(255)	
surname	Powers	Alfanumérico	VARCHAR(255)	
phone	(0111) 367 0184	Alfanumérico	VARCHAR(255)	
email	ante.blandit@outlook.edu	Alfanumérico	VARCHAR(255)	
birth_date	Mar 20, 2000	Alfanumérico, en este caso no está en patrón de DATE.	VARCHAR(255)	
country	Canada	Alfanumérico	VARCHAR(255)	
city	Rigolet	Alfanumérico	VARCHAR(255)	
postal_code	V6T 6M7	Alfanumérico	VARCHAR(255)	
address	154-5415 Auctor St.	Alfanumérico	VARCHAR(255)	

TABLA TRANSACTIONS				
COLUMNA	**DATOS A INSERIR**	**TIPOLOGIA**	**USADO**	**RELACCIÓN**
id	108B1D1D-5B23-A76C-55EF-C568E49A05DD	36 caracteres alfanuméricos.	VARCHAR(255)	**PRIMARY KEY**
card_id	CcU-2938	8 caracteres alfanuméricos.	VARCHAR(255)	**FOREING KEY**
business_id	b-2222	6 caracteres alfanuméricos.	VARCHAR(255)	**FOREING KEY**
timestamp	07/07/2021 17:43	Almacena data e hora	TIMESTAMP	
amount	293.57	Valores decimales con precisión exacta	DECIMAL (10,2)	
declined	0 ou 1	Valores binarios (0/1, verdadero/falso).	BOOLEAN	
product_ids	11, 13, 61, 29	Almacena múltiples IDs como texto	VARCHAR(255)	
user_id	275	Números enteros	INT	**FOREING KEY**
lat		Números decimales	FLOAT	
longitude	-178.860.353.536	Números decimales	FLOAT	

Creación del database y tablas. Iniciamos con las tablas de dimensiones:

Tabla db_company (dimensiones):

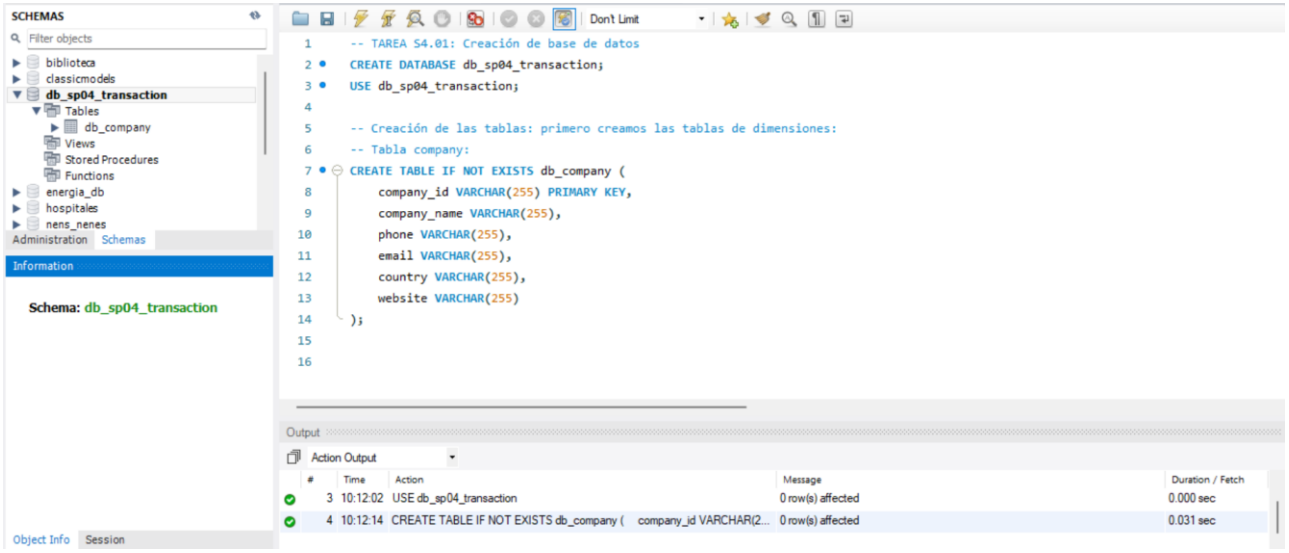


Tabla db_credit card (dimensiones):

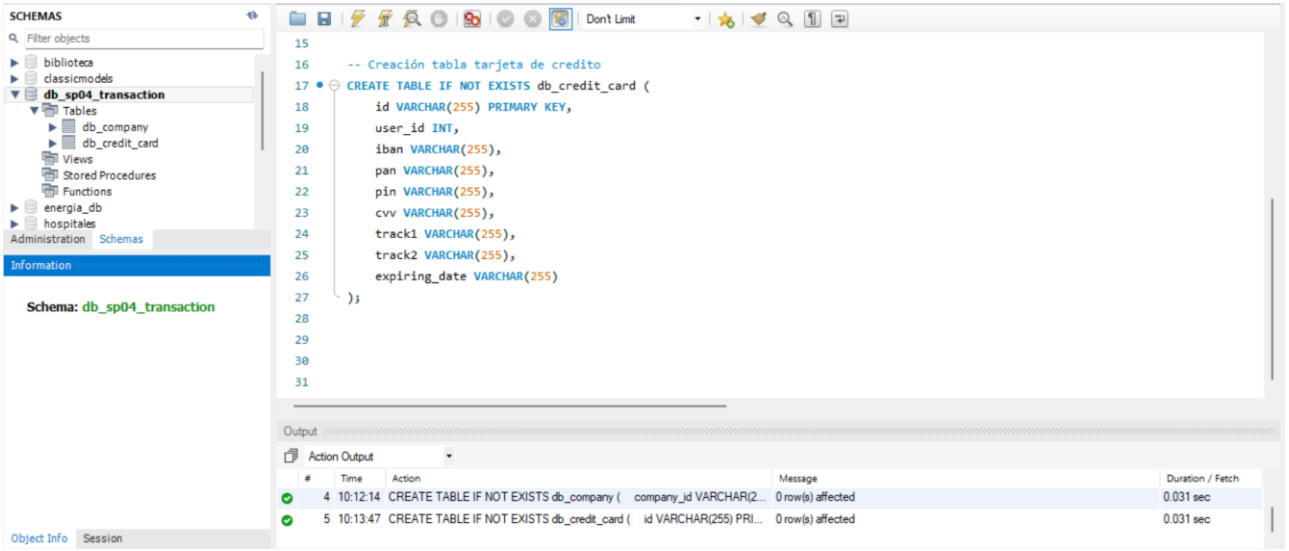
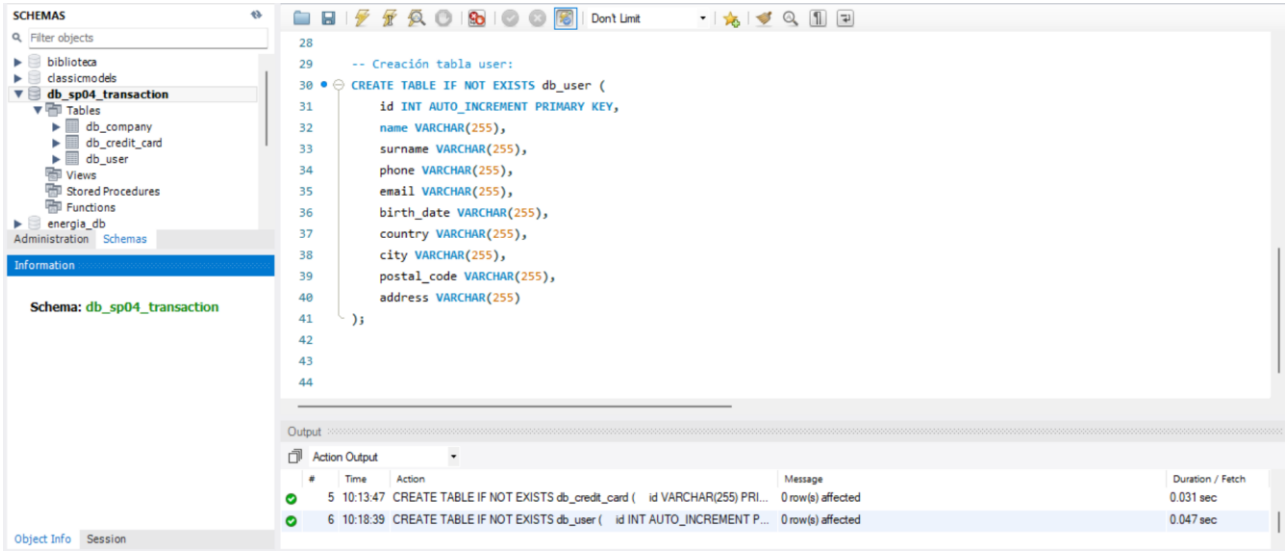
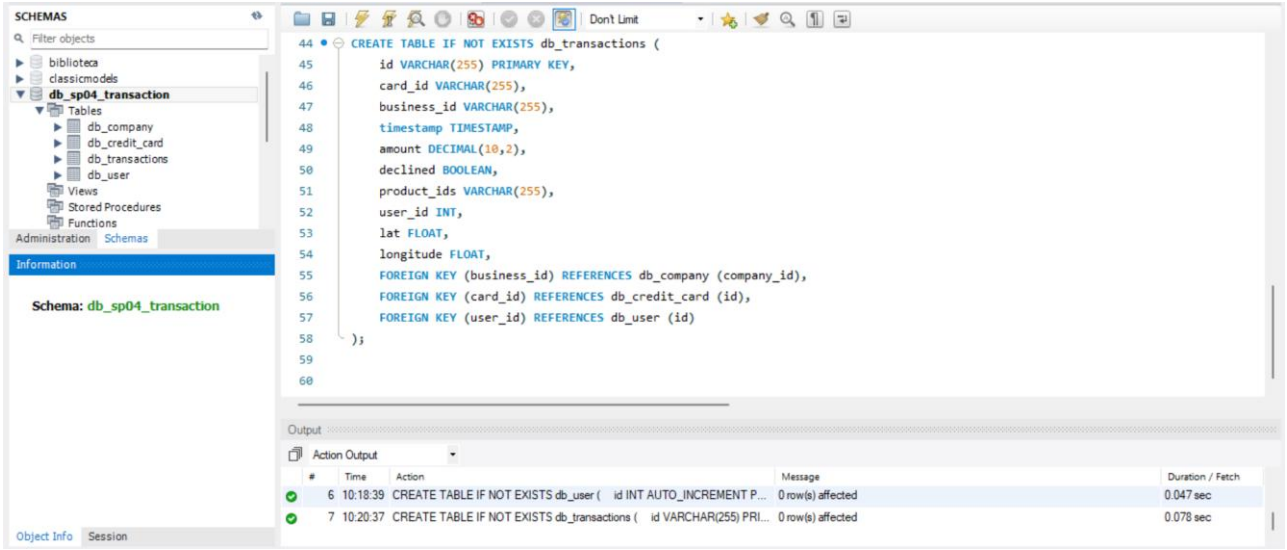


Tabla db_user (dimensiones):



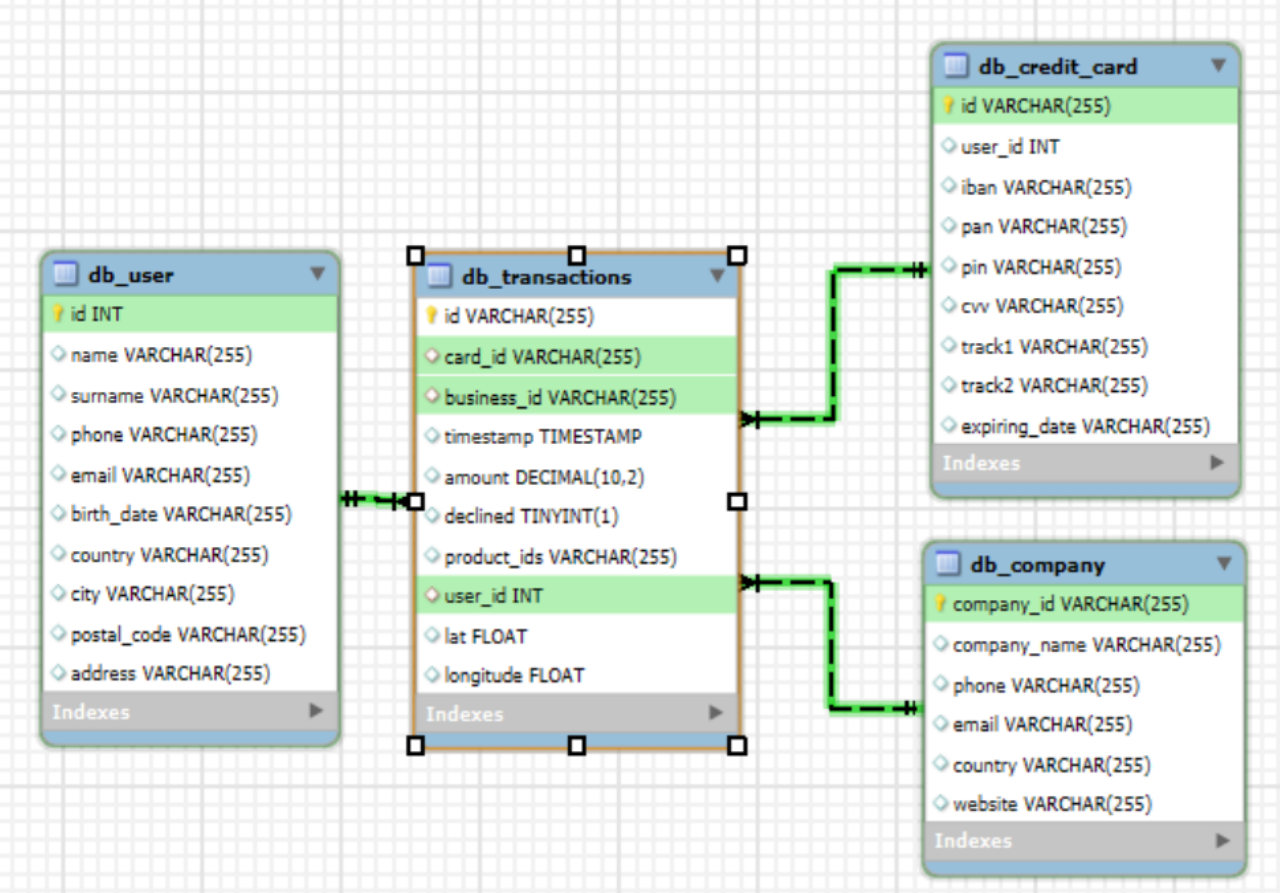
Tablas db_transactions (tabla de hechos):



Se ha creado un modelo en estrella, donde la tabla de hechos es db_transactions (1:N) y las tres dimensiones son:

- *db_credit_card (datos de las tarjetas):* un usuario puede tener muchas tarjetas de crédito.
- *db_user (datos de los usuarios, combinando tablas de UK, CA y USA):* un usuario puede realizar muchas transacciones.
- *db_company (datos de las compañías):* una empresa puede recibir muchas transacciones.

En este sentido, el modelo relacional se presenta de esta forma:



Como segundo paso, inserimos los datos fornecidos:

Primero, verificamos el formato de los datos, si están separados por coma o punto y coma:
Formato datos companies:

```
company_id,company_name,phone,email,country,website
b-2222,Ac Fermentum Incorporated,06 85 56 52
33,donec.porttitor.tellus@yahoo.net,Germany,https://instagram.com/site
```

Formato datos credit_cards:

```
id,user_id,iban,pan,pin,cvv,track1,track2,expiring_date
CcU-2938,275,TR301950312213576817638661,5424465566813633,3257,984,%B8383712448554646^WovsxejDpwiev^8604114?7,%
B7653863056044187=8007163336?3,10/30/22
```

Formato datos users:

```
id,name,surname,phone,email,birth_date,country,city,postal_code,address
151,Meghan,Hayden,0800 746 6747,arcu.vel@hotmail.ca,"Jul 2, 1980",United Kingdom,Tullibody,A1Y 3TC,Ap #432-4493
Aliquet Rd.
```

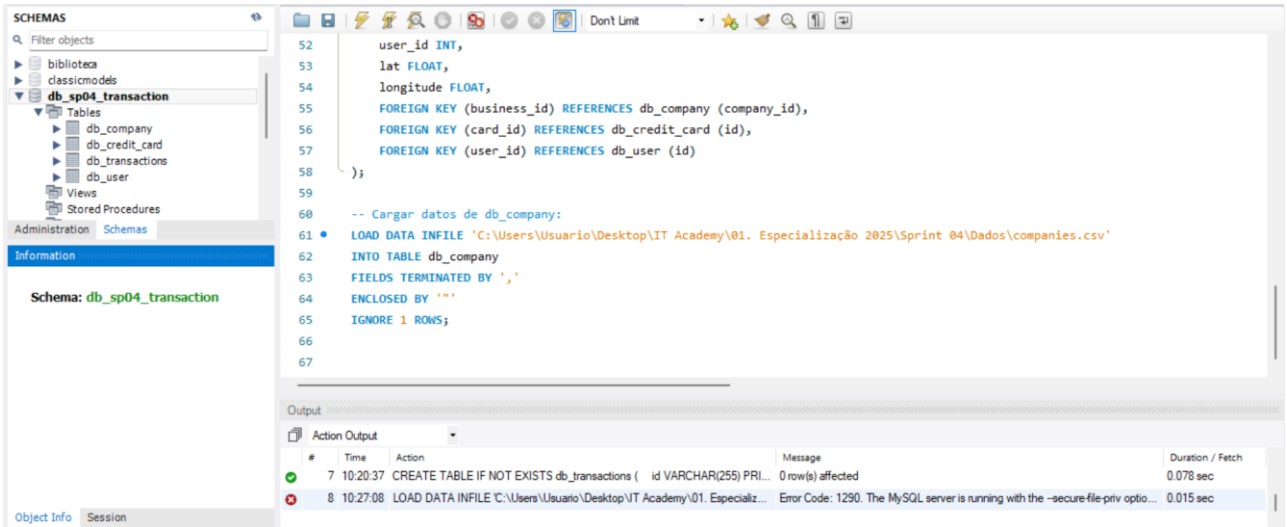
Formato datos transactions:

```
id;card_id;business_id;timestamp;amount;declined;product_ids;user_id;lat;longitude
108B1D1D-5B23-A76C-55EF-C568E49A05DD;CcU-2938;b-2222;2021-07-07
17:43:16;293.57;0;59;275;83.7839152128;-178.860353536
7DC26247-20EC-53FE-E555-B6C2E55CA5D5;CcU-2945;b-2226;2022-02-04 15:52:56;312.5;0;71,
41;275;58.9367181312;-76.8171099136
```

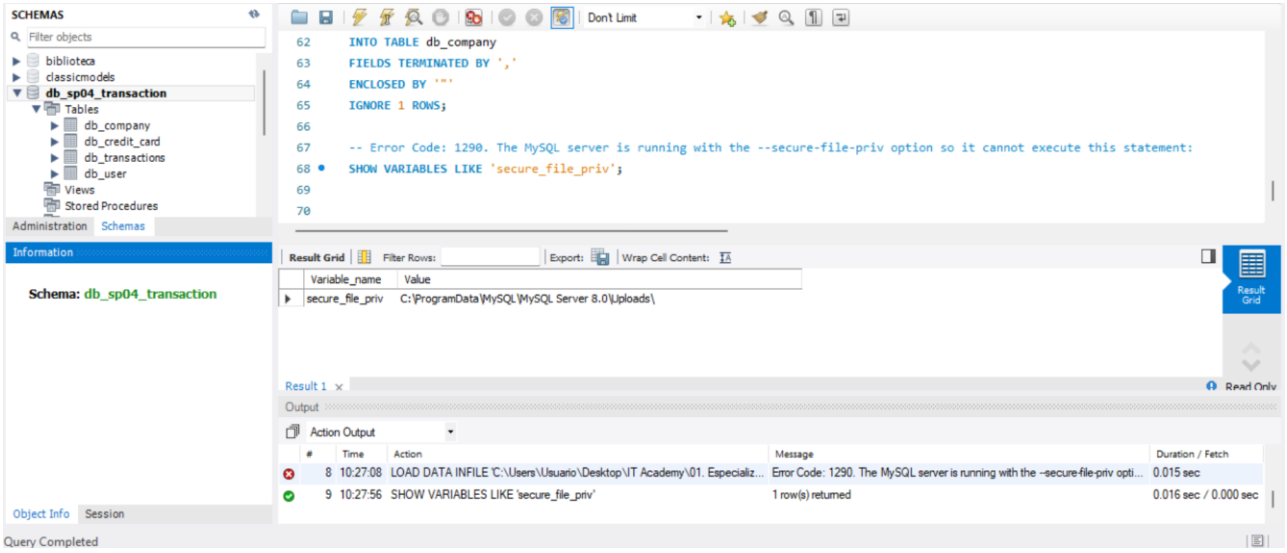
Para importar los datos de un Excel, en este caso archivo csv, utilizamos el siguiente comando (MySQL Manual):

1. LOAD DATA INFILE C:\Users\Usuario\Desktop\IT Academy\01. Especialização 2025\Sprint 04\Dados***tabla.csv'.*** **--**Esta línea carga los datos directamente un archivo csv.
   ***CUIDADO: Tiene que ser / y no ***
2. INTO TABLE ***nombre table***--Aquí como en la inserción normal, indicamos el nombre de la tabla donde irán los datos.
3. FIELDS TERMINATED BY ','. – Aquí definimos que los campos están separados (delimitador de campos) por coma (,) para las tablas companies, credit_cards y users y por punto y coma (;) para la tabla transactions.
4. ENCLOSED BY '"' – Esto explica que los campos pueden estar con doble comillas. Usual utilizarlo cuando estamos cargando datos csv.
5. IGNORE 1 ROWS; -- Por fin, pedimos para que ignore la primera fila, que es nombre de cada columna.

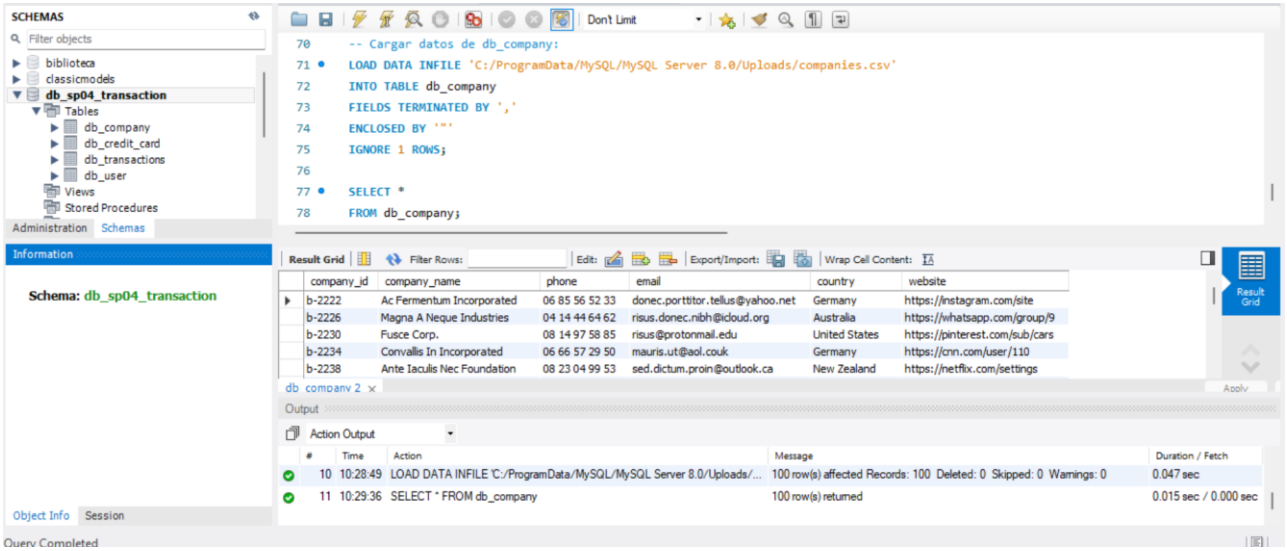
Inicialmente, como primero intento para inserir los datos, salió el error:
Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement:



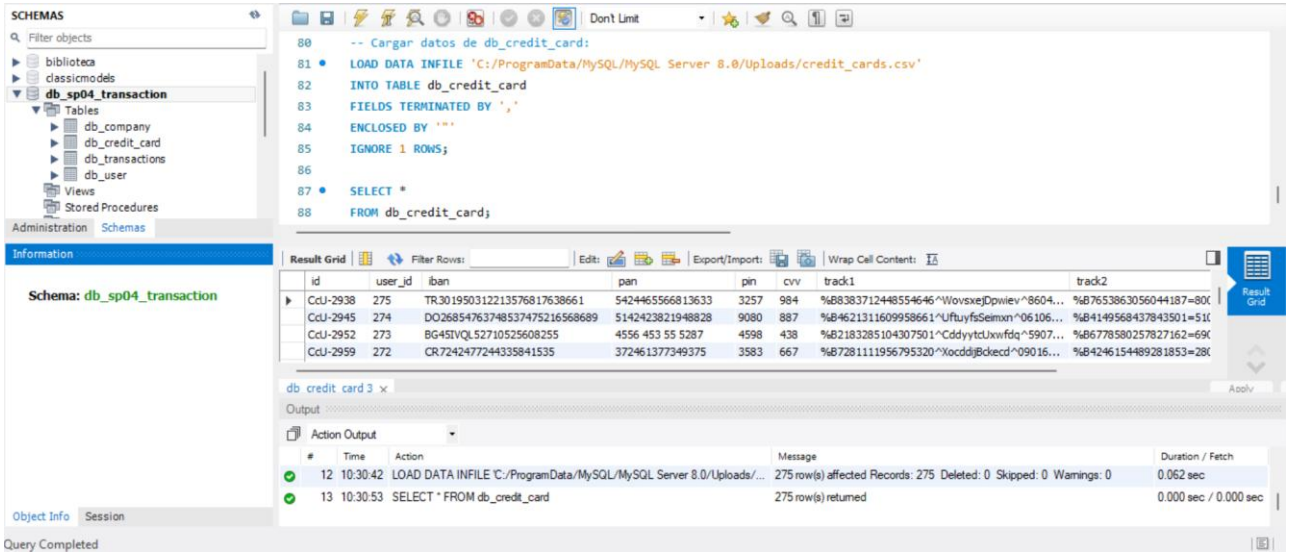
En este caso, se ha buscado el camino default de MySQL para salvar los archivos que serán utilizados:



Inserción de los datos de la tabla db_company:

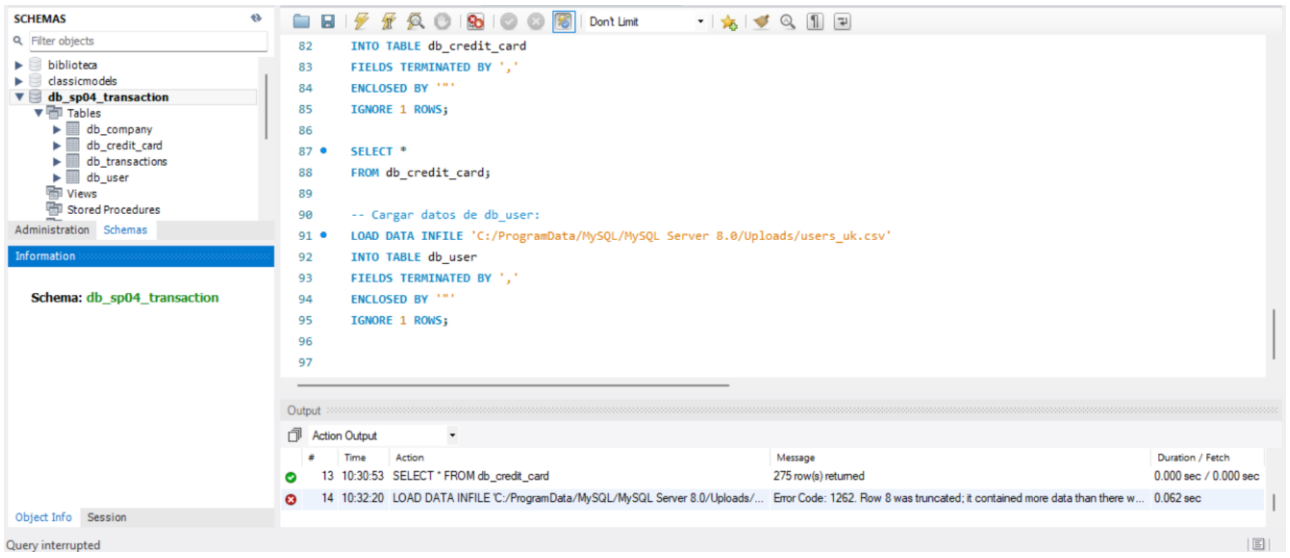


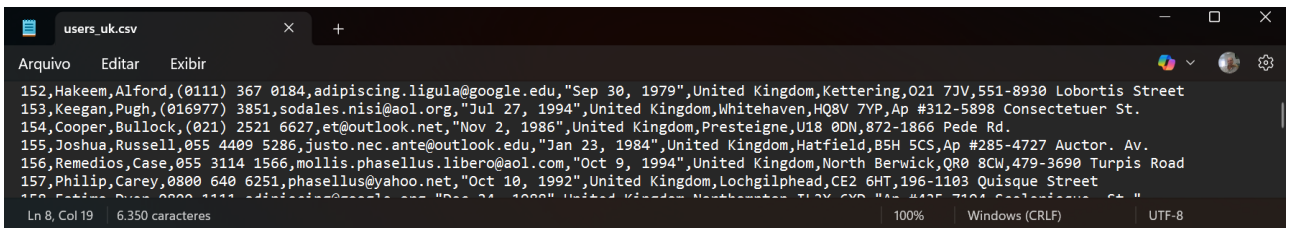
Inserción de los datos de la tabla db_credit_card:



Inserción de los datos de la tabla db_user:

Aquí, nos presentó el siguiente error: *Error Code: 1262. Row 8 was truncated; it contained more data than there were input columns.*

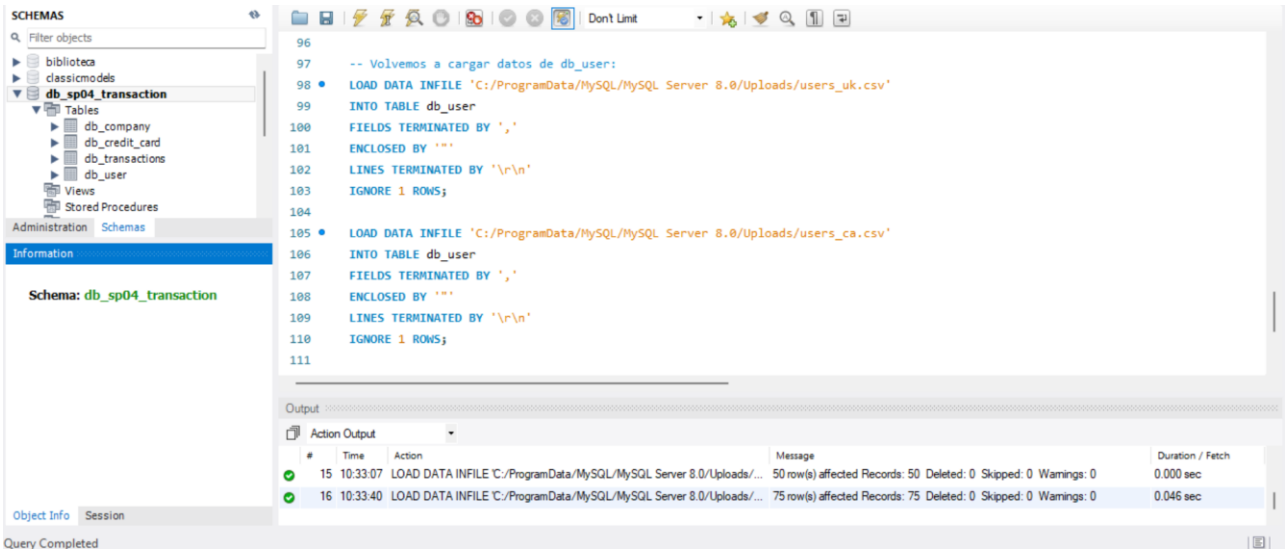




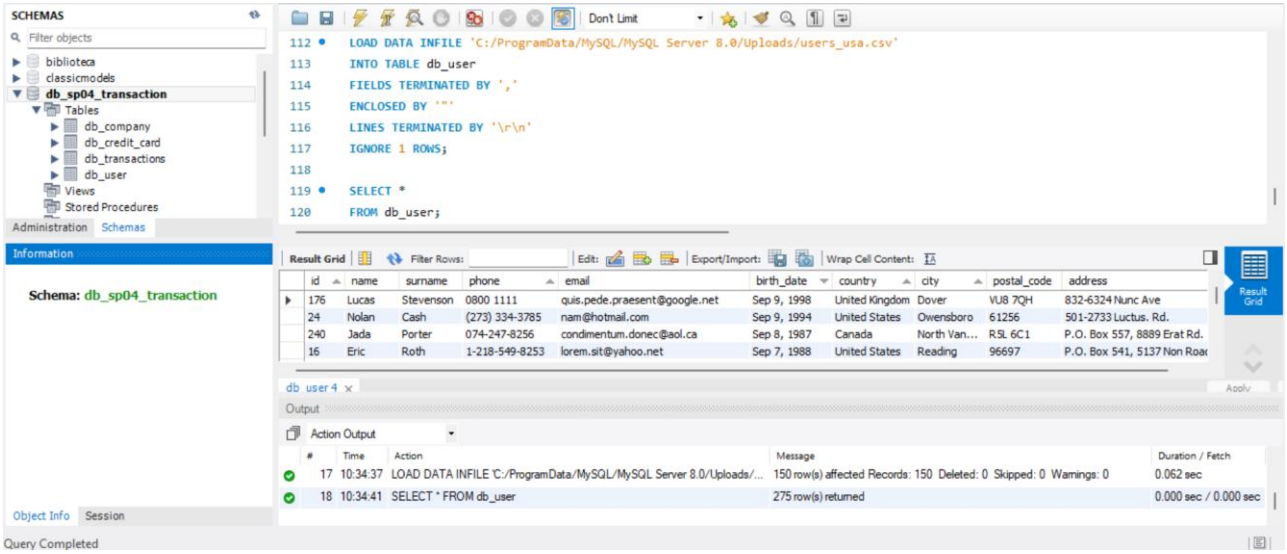
El error ocurre cuando una línea del archivo contiene más columnas de lo que esperado por la tabla, o cuando los archivos fueron creados con Windows (en este caso, como en la figura de arriba, Windows (CRLF)). Aís, es necesario especificar que la importación de los datos tenga las líneas que terminen con \r\n**."**

- **LINES TERMINATED BY '\r\n'** – Combinación de retorno de carro + avance de línea.

Inserción de los datos user_uk y ca:

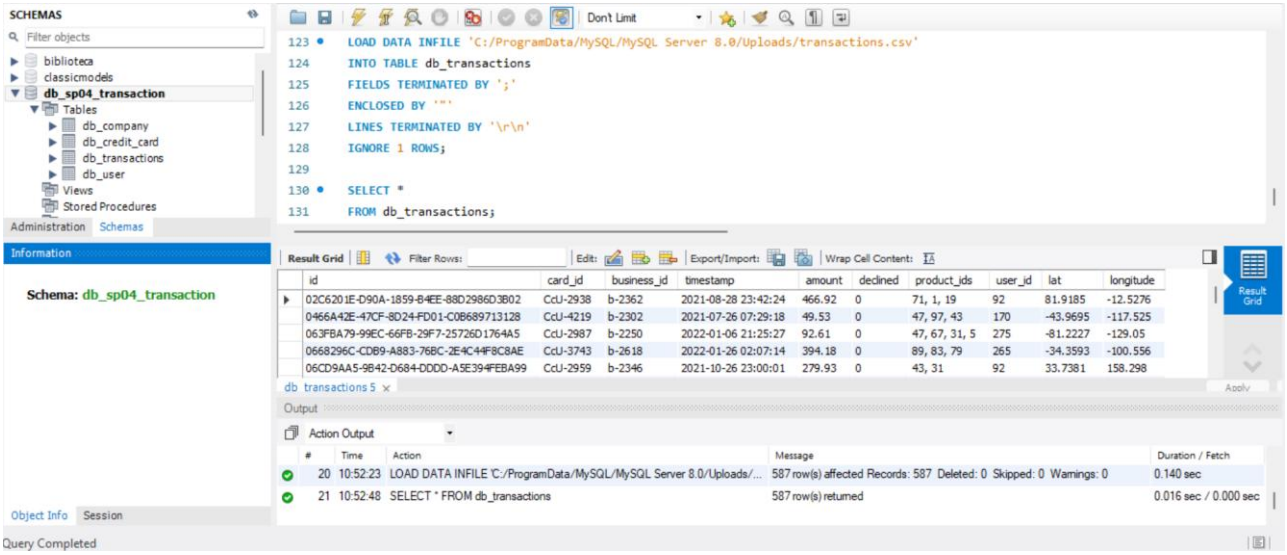


Inserción de los datos user_usa y verificación de la carga:



Mirando a la columna country, verificamos que los 3 países se han insertado correctamente. Totalizando en 275 usuarios.

Por último, inserimos los datos de las db_transactions.
Como los datos del Excel, en esta tabla en especifico está delimitado por ';', modificamos el FIELDS TERMINATED BY ','por FIELDS TERMINATED BY ';':

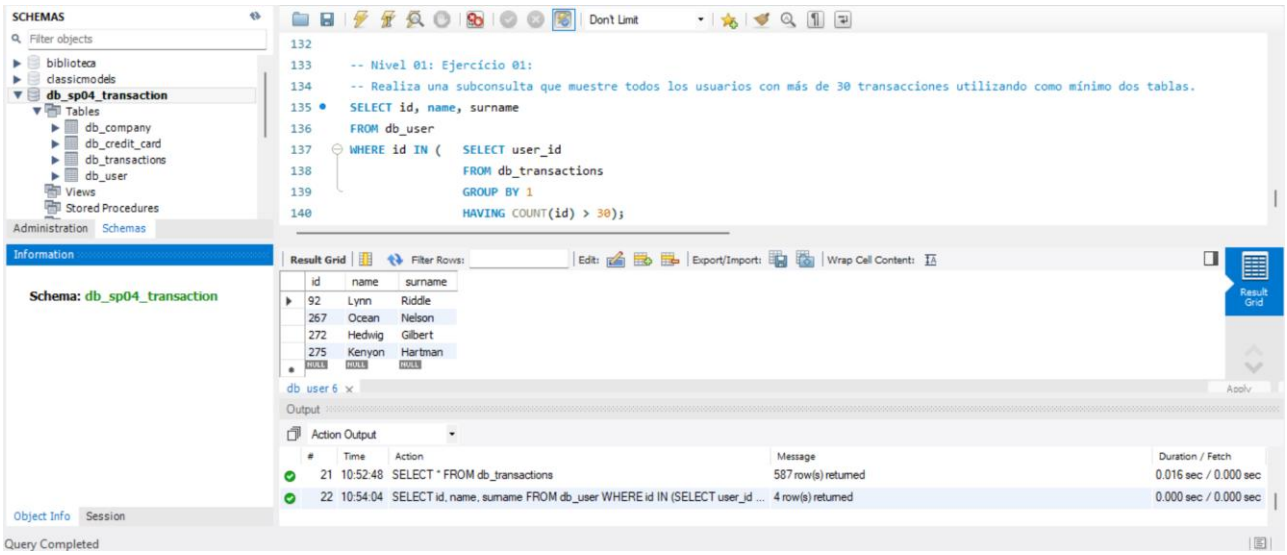


Tenemos en total 587 transacciones realizadas.

EJERCICIO 01

Realiza una subconsulta que muestre todos los usuarios con más de 30 transacciones utilizando como mínimo dos tablas.

Para este ejercicio, si utilizó la tabla db_user (campos id, name y surname) y la tabla db_transactions. La subconsulta se realizó en la cláusula **WHERE**, haciendo la combinación de las dos tablas, siendo utilizado el **HAVING** para filtrar los usuarios con más de 30 transacciones.

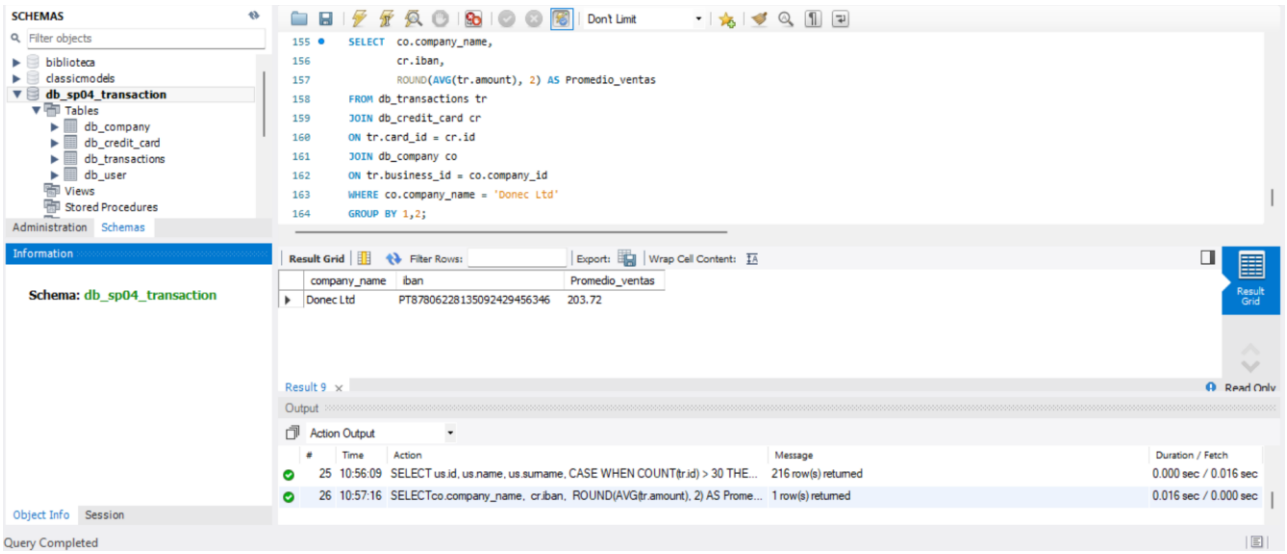


Tenemos a 4 usuarios que hicieron más de 30 transacciones.

EJERCICIO 2

Muestra el promedio del monto por IBAN de las tarjetas de crédito hechas a la compañía Donec Ltd, utiliza como mínimo 2 tablas.

Para esta consulta, se ha hecho una JOIN ente las tres tablas (db_company, db_transactions, db_credit_card) para llegar al resultado esperado. En la consulta, pedimos el AVG del amount y mediante a un filtro WHERE, filtramos la empresa Donec Ltd (en la tabla db_transactions como el business_id = b-2242):

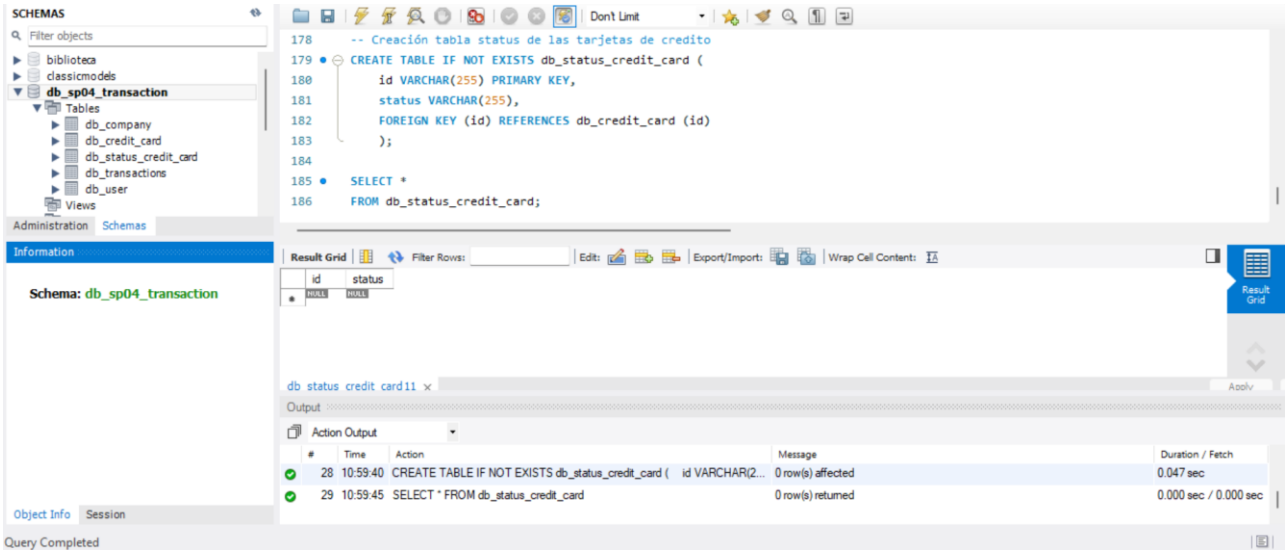


Como resultado tenemos solo un IBAN, que ha realizado dos compras de 354,61 y 42,82, totalizando un promedio de 203,72.

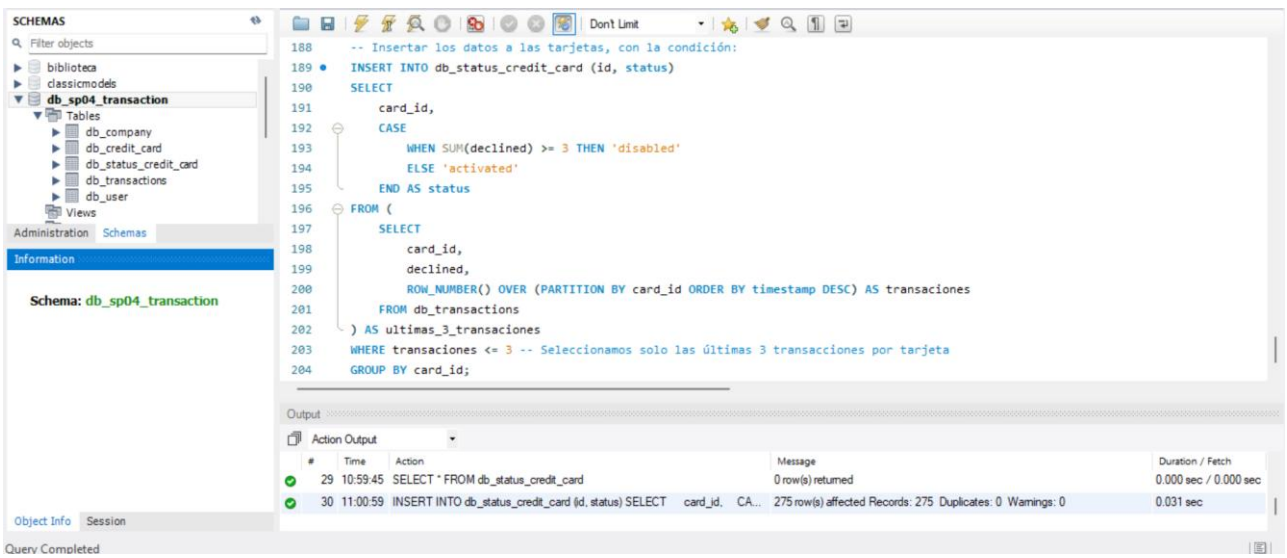
NIVEL 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas y genera la siguiente consulta:

Creamos una tabla db_status_credit_card:



Para hacer la creación de los datos de la columna status, fue creada una condición para que pudiera rellenar los datos, de acuerdo con el enunciado: *refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas.*
Esta tarea, fue particularmente complicada, y después de muchos intentos, solucionamos con este código (revisado por peer):

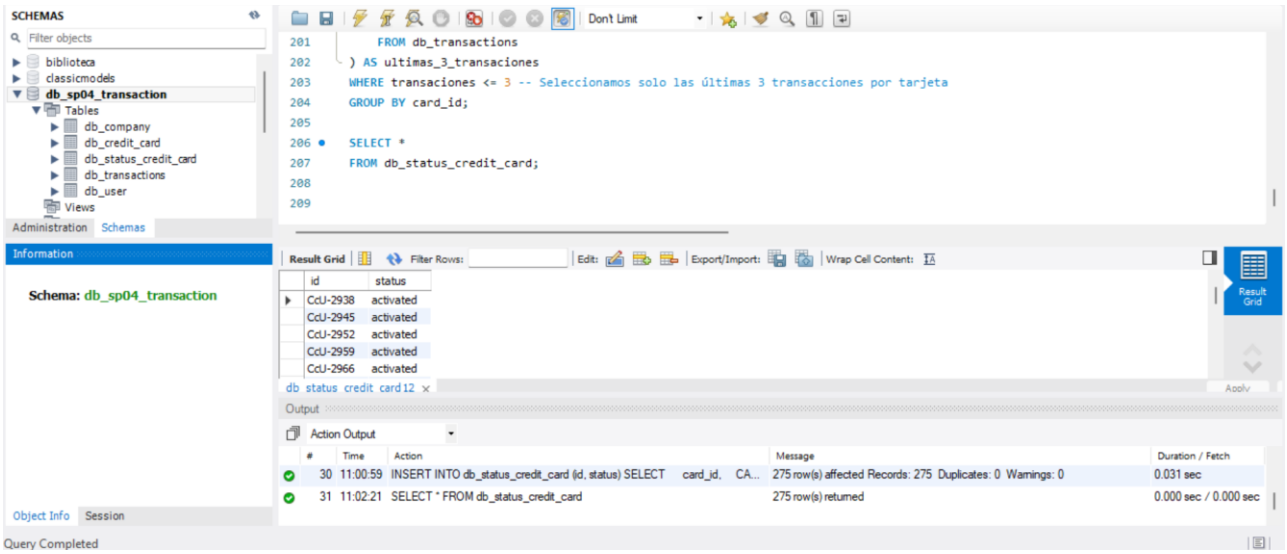


Inicialmente, utilizamos el **INSERT INTO** para insertar los datos en la tabla **db_status_credit_card**, en sus dos columnas: **id** y **status**. Para esto, creamos una condición, relacionado con las transacciones de cada tarjeta. Con el **CASE WHEN** creamos la condición, con el **SELECT** buscamos las informaciones de la tabla db_transactions, con una subconsulta que mira cada transacción y organiza de la más reciente de cada tarjeta usando la función **ROW_NUMBER()**. Esto numera las transacciones de cada tarjeta (**PARTITION BY card_id**) empezando de más reciente (**ORDER BY timestamp DESC**).
Después el código filtra solo las tres últimas transacciones de cada tarjeta con la condición **WHERE transaciones <= 3**. Agrupamos por el card_id, usando la función **SUM(declined),** para contar cuantas estas últimas (utilizando el timestamp, para verificar las fechas) transacciones fueron declinadas. Se este sumatorio es mayor o igual a 3 el estatus será **'disable'**, si no es así será **'activated'.**

- **ROW_NUMBER()** – assigna um número de fila único.
- **OVER(ORDER BY columna)** – Indica cómo deben ordenarse los datos para asignar los números del ROW_NUMBER. (En este caso, ordenamos por timestamp, así tenemos las últimas 3 transacciones)
- **PARTITION BY** – divide los datos, para dividir los ids de la columna product_ids de db_transactions.

Fuentes: learnsql, stackoverflow, sqlshack

Hecha la creación de la tabla y añadido los campos, verificamos si la tabla tiene la información correcta:

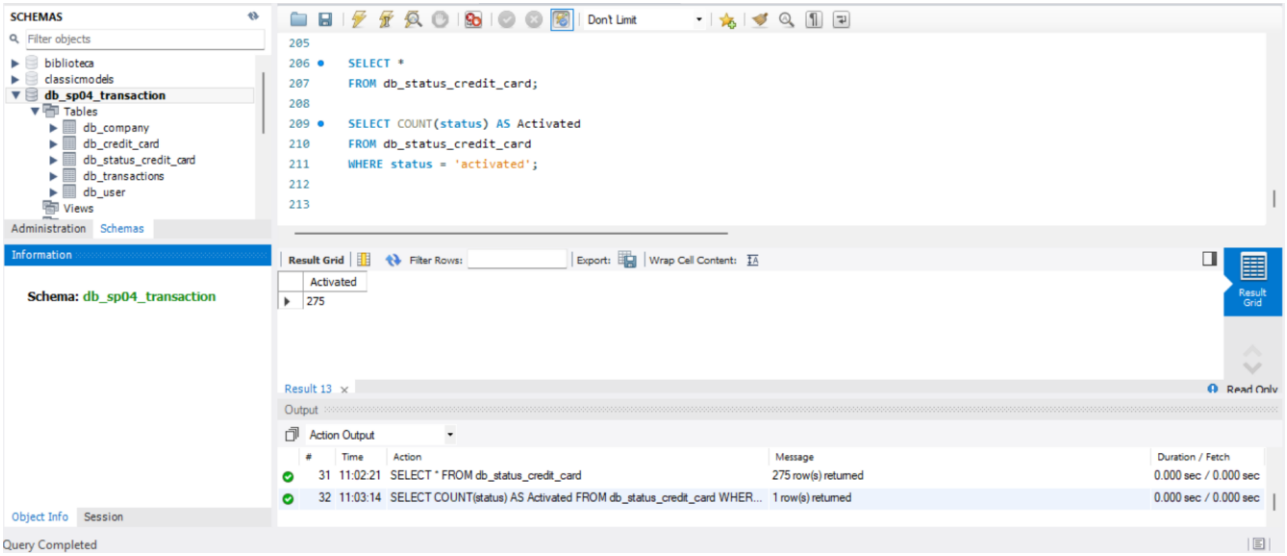


Muestra las 275 tarjetas del banco de dados, ahora con el estatus actualizado.

EJERCICIO 01

¿Cuántas tarjetas están activas?

Para esta consulta, si ha hecho el contaje de las tarjetas que tienen el estatus 'activated'



Como resultado, tenemos que todas (275) las tarjetas están activas (activated).

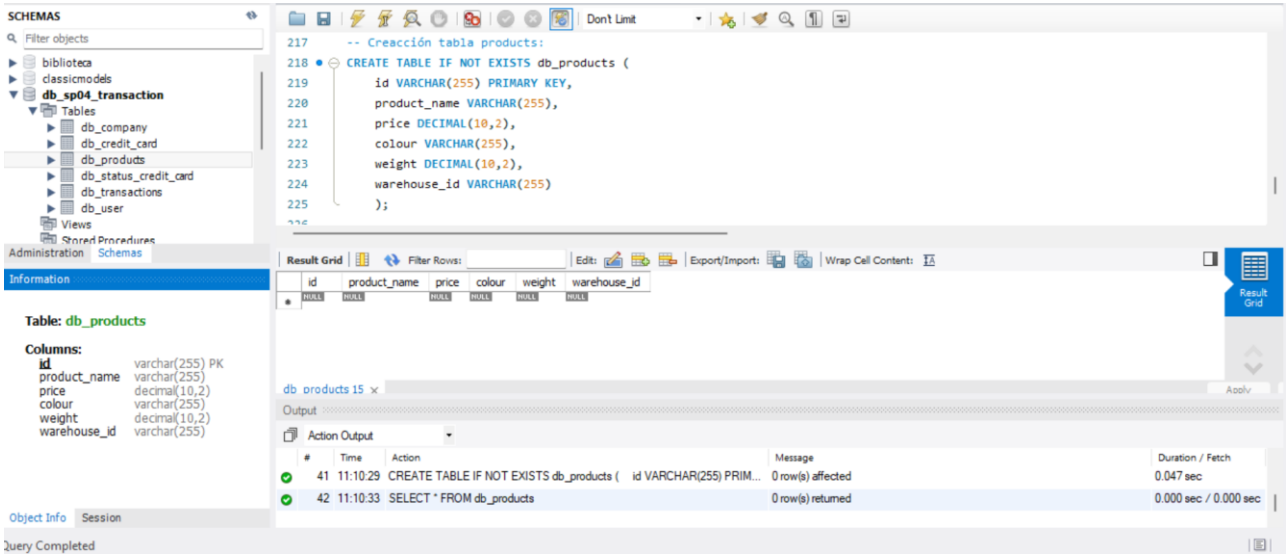
NIVEL 3

Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes producto_ids. Genera la siguiente consulta:

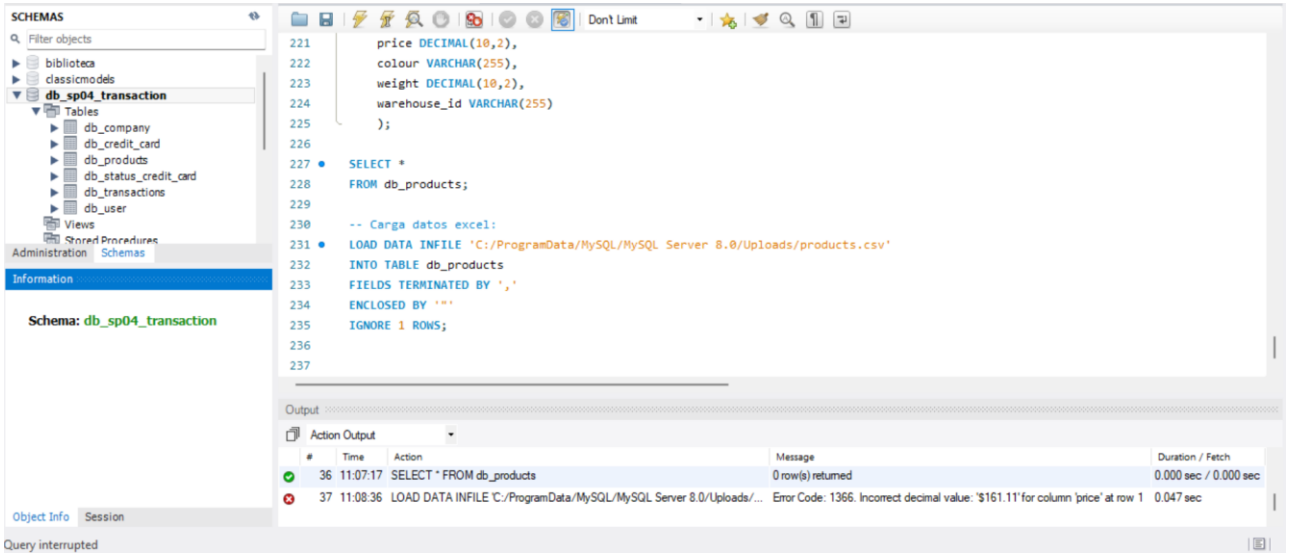
Inicialmente, revisamos los datos de la tabla products:

TABLA PRODUCTS				
COLUMNA	**DATOS A INSERIR**	**TIPOLOGIA**	**USADO**	**RELACCIÓN**
id	1	Entero	VARCHAR(255): mismo que en db_transactions.	**PRIMARY KEY**
product_name	Direwolf Stannis	caracteres alfanuméricos.	VARCHAR(255)	
price	$161.11	Decimal, presenta los precios	DECIMAL(10, 2)	
colour	#7c7c7c	caracteres alfanuméricos.	VARCHAR(255)	
weight	1.5	Float o decimal	DECIMAL(10, 2)	
warehouse_id	WH-4	caracteres alfanuméricos.	VARCHAR(255)	

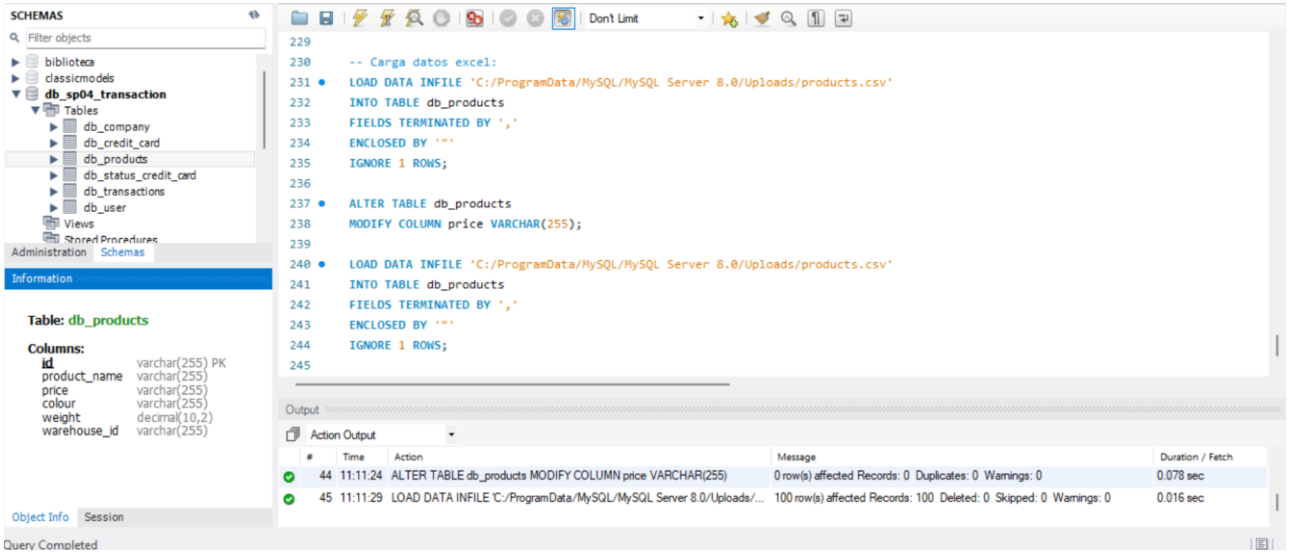
Creación de la tabla db_products:



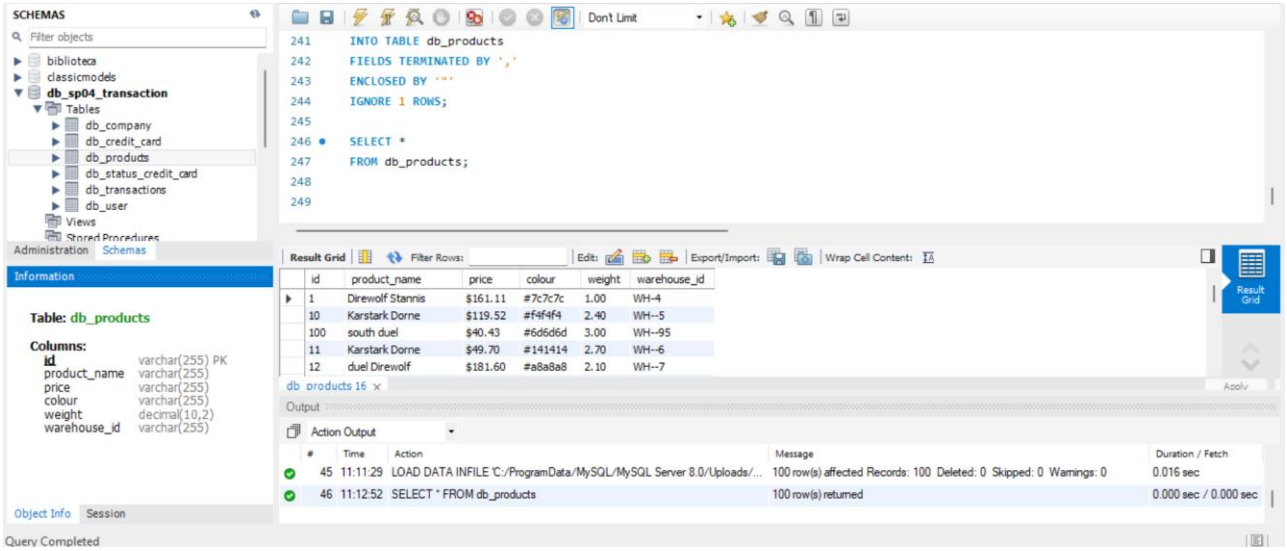
Añadimos los datos del Excel e inicialmente nos da el error:
Error Code: 1366. Incorrect decimal value: '$161.11' for column 'price' at row 1.



En este caso, modificamos la tabla, para que los datos de la columna Price sean VARCHAR(255).



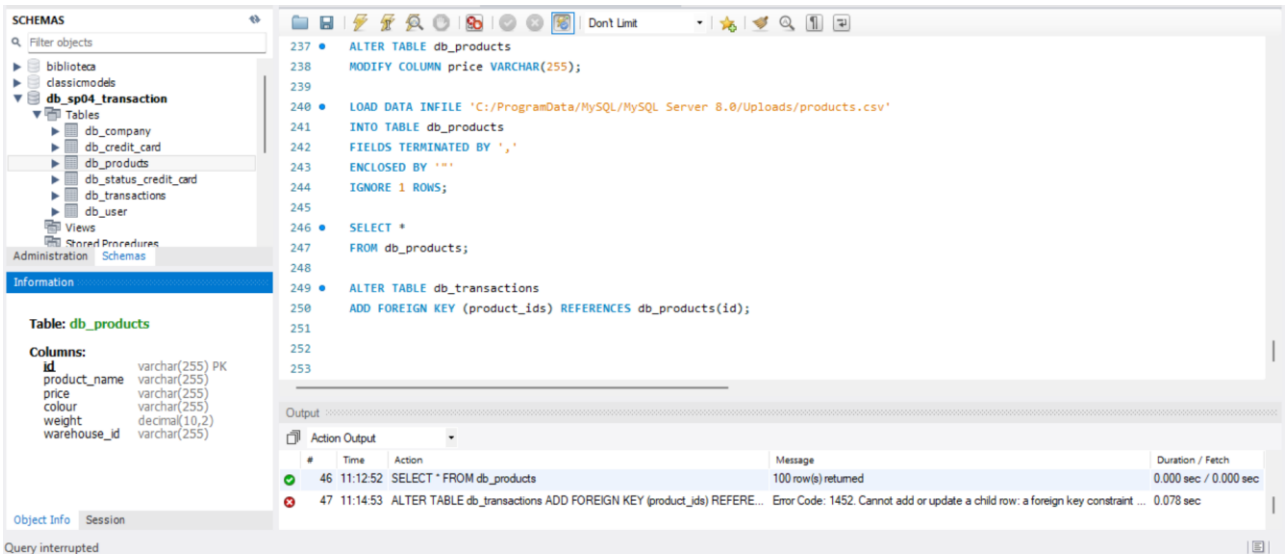
Verificamos los datos de la tabla:



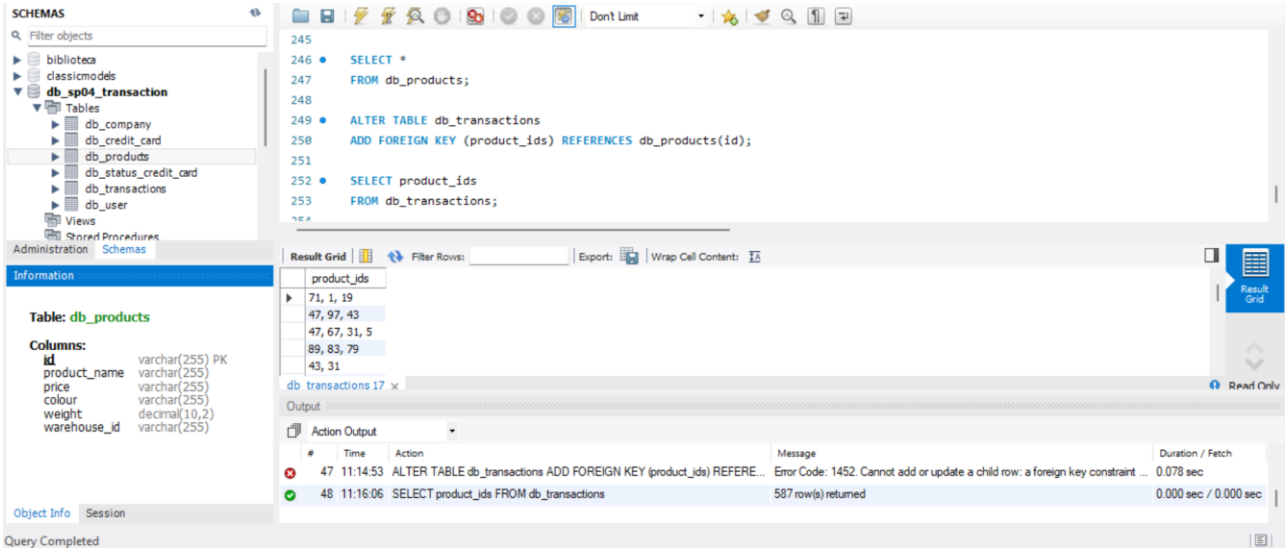
Tenemos 100 productos registrados en el database.

Para hacer la consulta, falta crear la clave foránea en la tabla db_transactions para que las dos tablas puedan relacionarse:

En este caso nos dio en siguiente error:
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`db_sp04_transaction`.`#sql-10cc_9`, CONSTRAINT `db_transactions_ibfk_4` FOREIGN KEY (`product_ids`) REFERENCES `db_products` (`id`))

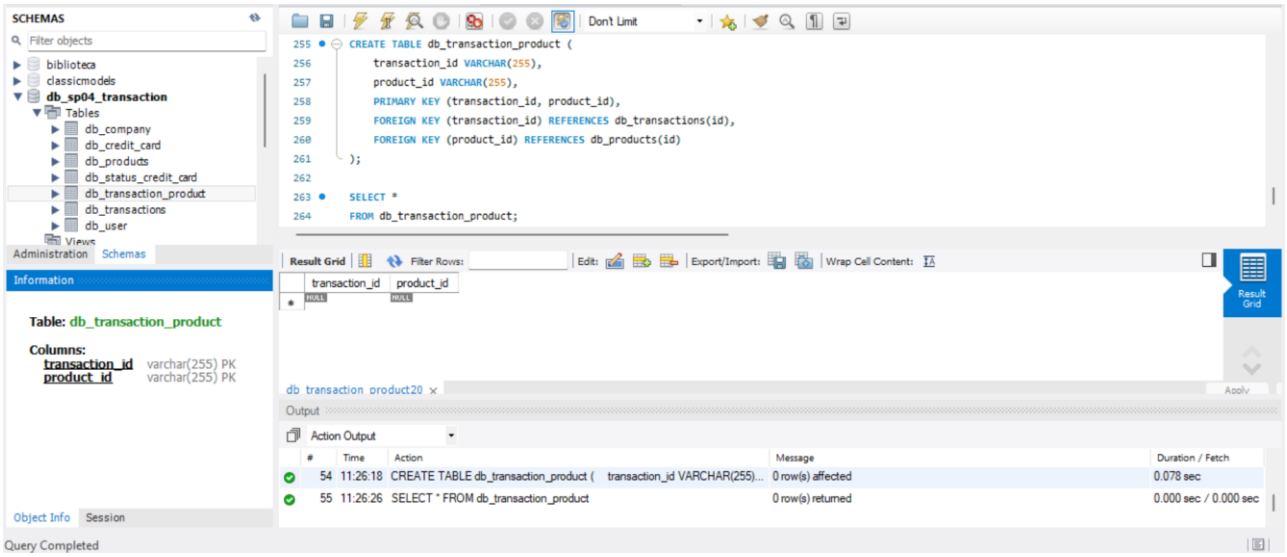


Mirando los id en las dos tablas, percibimos que en db_transactions, los products_ids está en el formato: '83, 97, 23, 71'. Lo que ya habíamos visto en fase inicial de la creación del database. En este caso, no es posible crear la clave foránea, visto que existe más de un id de producto por transaction.



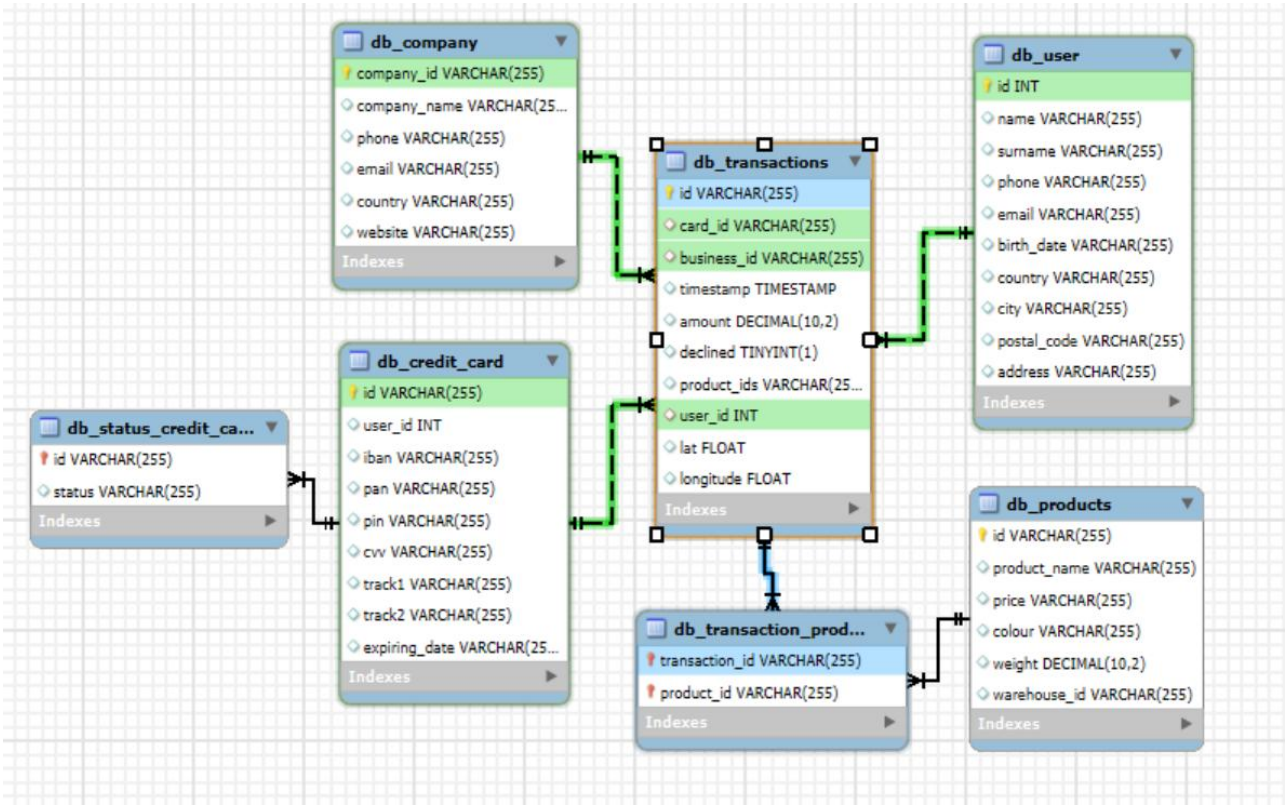
Para solucionarlo, se crea una nueva tabla (puente), con los datos de los productos y su respectiva transacción, en formato único, para eliminar los ids múltiplos en product_ids, de la tabla db_ transactions.

Creación de la tabla db_transaction_product:

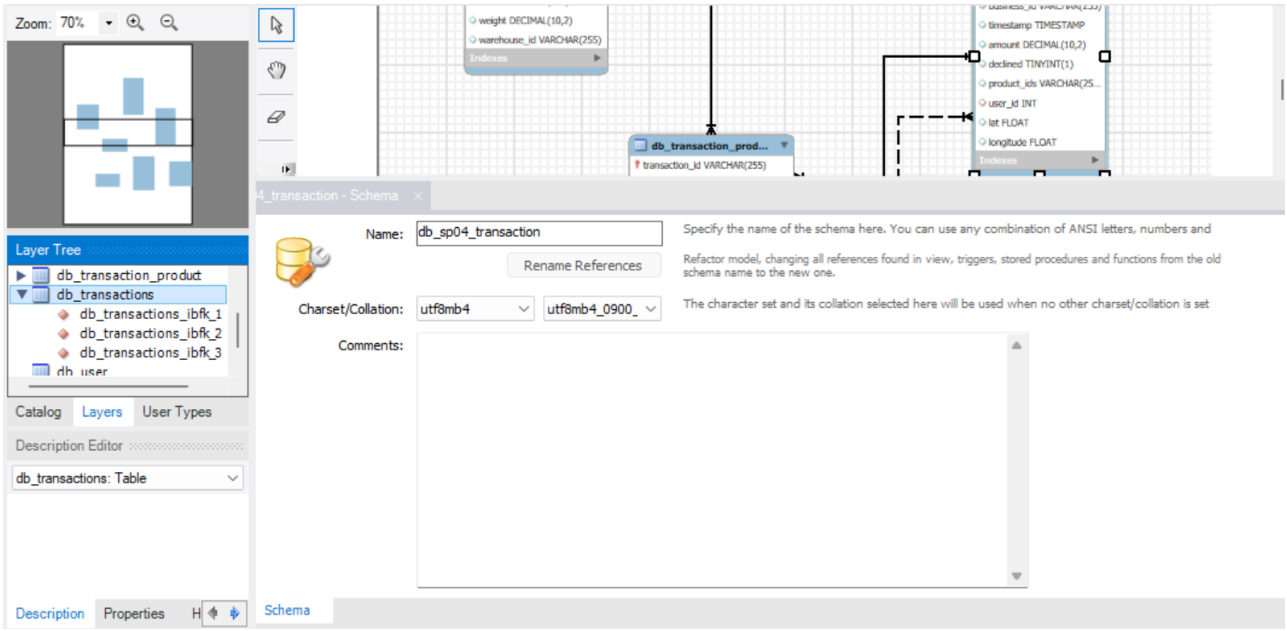


El modelo relacional sigue con la tabla de hechos es db_transactions (1:N), con las dimensiones users, credit_card y companies. Luego, se ha creado dos tablas puentes, para verificar el estatus de las tarjetas (db_status_credit_card) y una tabla puente (db_transaction_product) para verificar los

ids de los productos, relacionada directamente entre db_transactions y db_products. Este es el modelo finalizado:



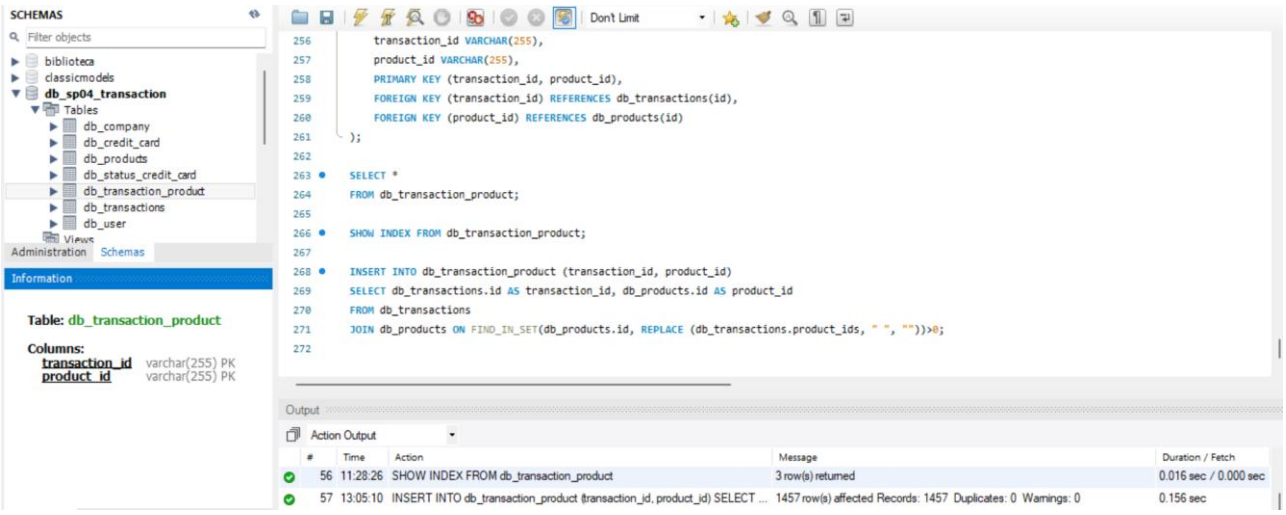
Verificamos las claves foráneas del modelo:



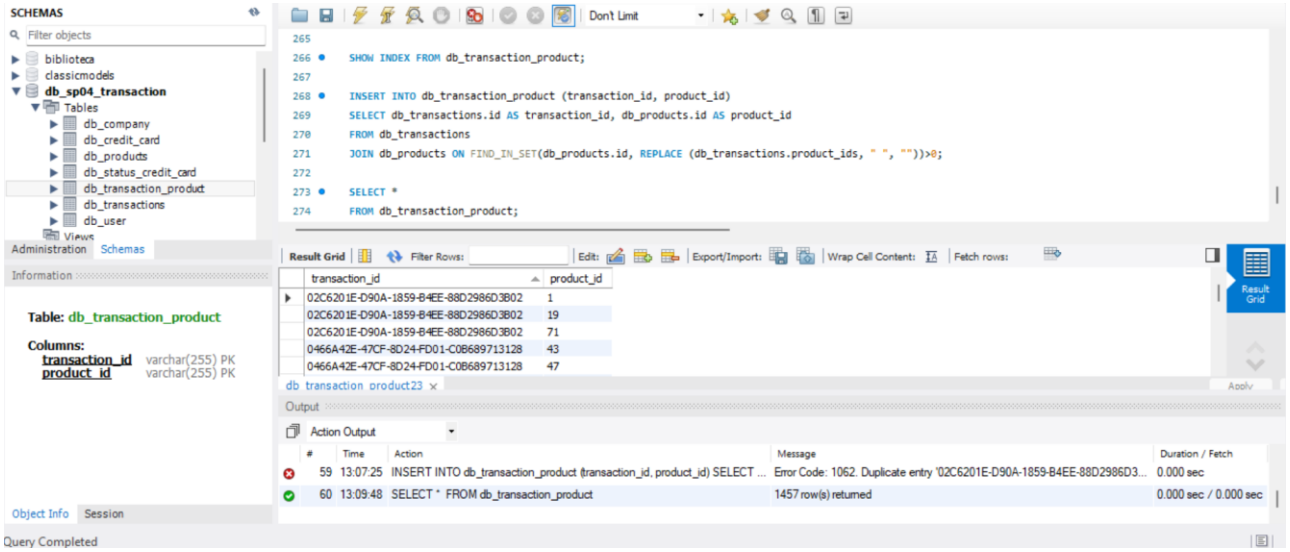
Por último, para insertar los datos y organizar los ids de productos de db_transactions para que sean únicos, se ha visto que existen muchas posibilidades. Inicialmente, se ha probado el STRING_SPLIT, pero solo funciona en SQL Server y no en MySQL. (fuente: Microsoft, youtube). En la peer revisión, si ha visto las funciones WITH RECURSIVE, TRIM y SUBSTRING INDEX (learnsql, galileu, MySQLManual, MySQLManual-substring).

Por fin, se ha decidido utilizar la función FIND_IN_SET y REPLACE (MySQLTutorial, MySQLManual), que pareció más sencilla, de esta forma se ha creado este código:

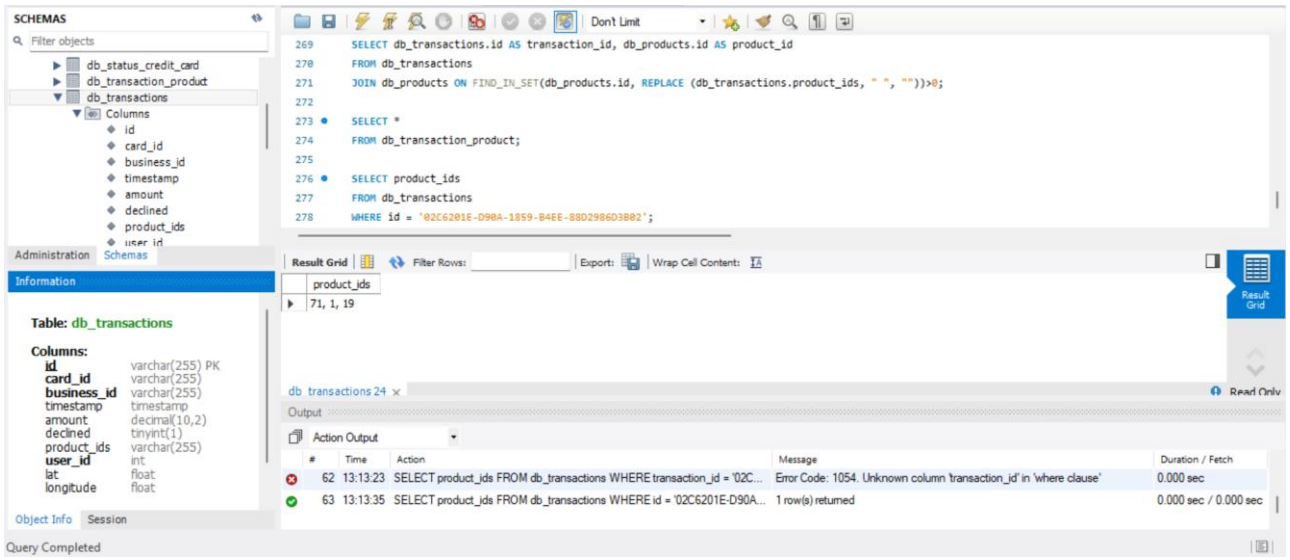
- **INSERT TO** db_transaction_product (transaction_id, product_id): comando para insertar el resutado en las columnas (transaction_id, product_id) de la nueva tabla (db_transaction_product).
- **SELECT** db_transactions.id AS transaction_id, db_products.id AS product_id: indicamos las columnas de las tablas db_transactions y db_products, donde queremos que se inserten los datos a la nueva tabla (db_transaction_product).
- **FROM** db_transactions: especificamos cual es la tabla principal para sacar los datos.
- **JOIN** db_products: unimos las dos tablas, con la condición a seguir
- **ON FIND_IN_SET(db_products.id, REPLACE(db_transactions.product_ids, " ", ""))>0:** db_transactions.product_ids es la columna que tiene multiplos ids, separados por coma. El " ", "" quita los espacios en blanco dentro de la string. La función FIND_IN_SET verifica si el id del producto está presente en la lista product_ids de la transacción. La función devuelve un valor mayor que 0 si el producto es encontrado, o 0 si no es encontrado. El >0 asegura que solo se seleccionen las filas donde el producto existe en la lista de product_ids.



Ahora verificamos que los ids, hayan sido ajustados de forma correcta:



Verificamos la transacción: '02C6201E-D90A-1859-B4EE-88D2986D3B02':

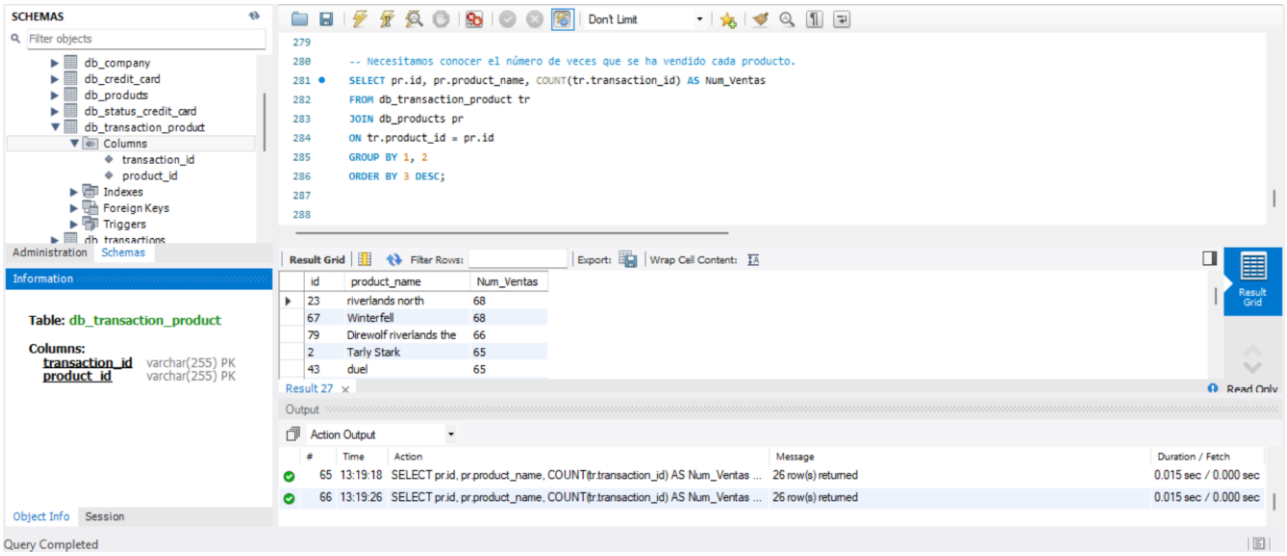


Aparecen los ids 1, 19 y 71, como en la imagen anterior, se entiende que la función FIND_IN_SET dividió los ids de la tabla db_transactions y luego el REPLACE reemplazó el id en una nueva línea con el mismo transaction_id, pero con solo un id de producto. En este caso especifico eran 3 productos diferentes y creo 3 lineas.

EJERCICIO 01

Necesitamos conocer el número de veces que se ha vendido cada producto.

Para esta consulta, con la nueva tabla db_transaction_product, hacemos el COUNT de transactions para encontrar cuantas veces se ha vendido cada producto.



Retornan 26 líneas, con el contaje de ventas de cada producto. Se percibe, que hay productos que no tienen ninguna transacción.