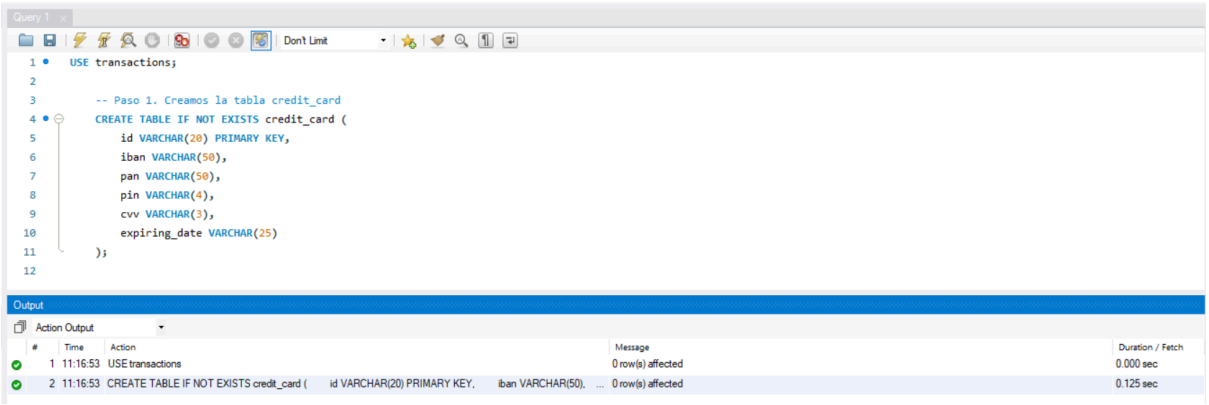
Nivel 1

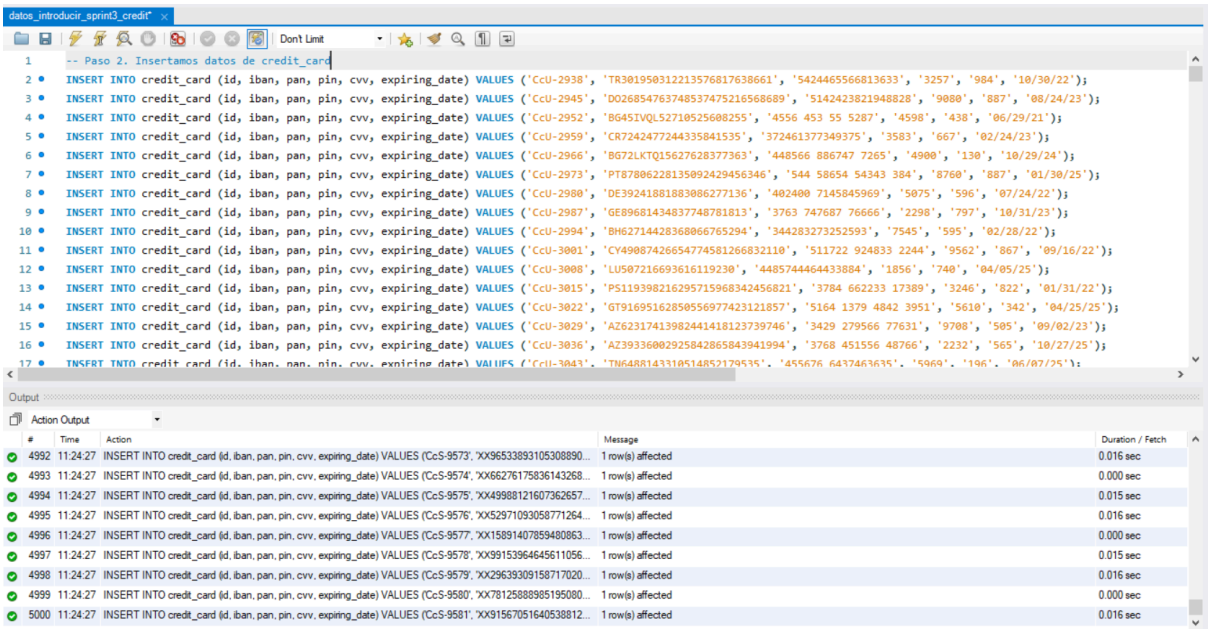
Ejercicio 1

Tu tarea es diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de forma única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después de crear la tabla será necesario que ingreses la información del documento denominado "datos_introducir_credit". Recuerda mostrar el diagrama y realizar una breve descripción del mismo.

Empiezo creando la tabla credit_card. Para decidir qué columnas debía contener, utilicé el archivo con los datos proporcionados y creé todos los campos necesarios. La columna id se define como clave primaria, ya que identifica de forma única cada tarjeta y permite establecer relaciones con otras tablas.

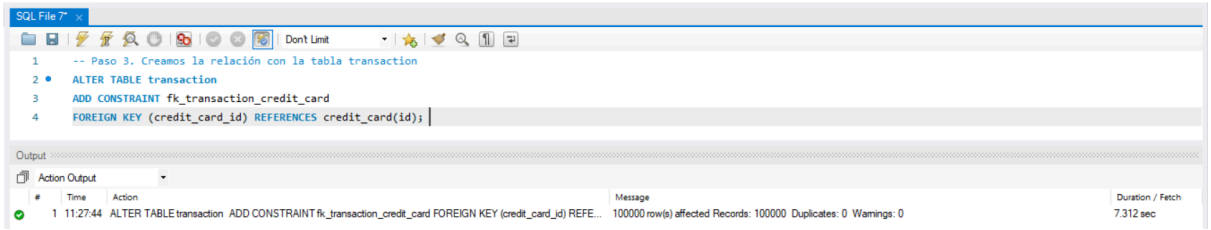


Después, inserté en la tabla todos los registros del archivo datos_introducir_credit para asegurar que la tabla no quedara vacía. Este paso es importante porque, si la tabla credit_card no tiene datos, no es posible crear correctamente la relación con la tabla transaction.



A continuación, añadí la relación entre credit_card y transaction.
El FOREIGN KEY establece que transaction.credit_card_id debe corresponder siempre a un valor existente en credit_card.id.
El constraint garantiza dos cosas:

- No se puede registrar en transaction una tarjeta que no exista en credit_card.
- No se puede eliminar una tarjeta de credit_card si todavía está asociada a alguna transacción.



El diagrama representa la relación entre 3 tablas: company, transaction y credit_card.

- “company” almacena la información de cada empresa (id, nombre, telefóno, email, país, página web)
- “credit_card” contiene los datos de cada tarjeta de crédito (id, iban, pan, pin, cvv y fecha de caducidad.

La tabla transaction actúa como una tabla de hechos (tabla puente), ya que contiene:

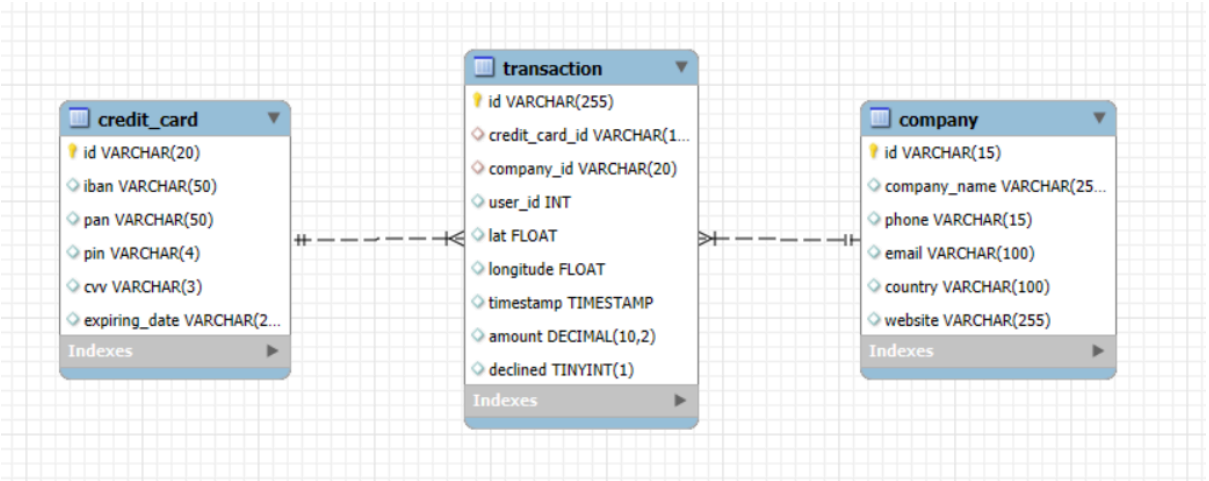
- ➢ company_id, que indica qué empresa realizó la transacción.
- ➢ credit_card_id, que indica con qué tarjeta se efectuó el pago.

Lo que permite saber qué empresa realizó la transacción y con qué tarjeta se pagó.

La relación del modelo queda así:

company (1) → (N) transaction (N) ← (1) credit_card

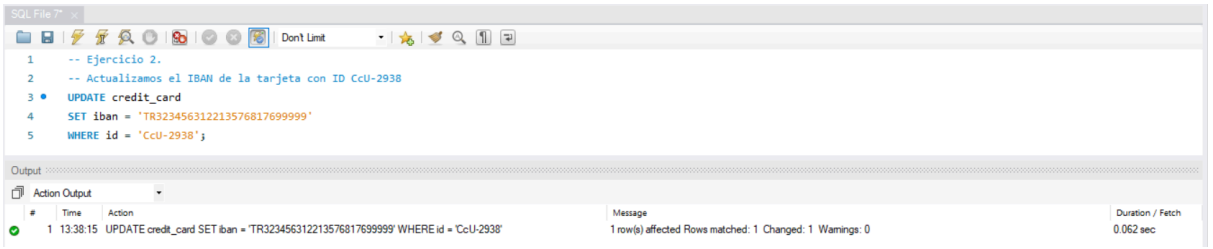
Es decir, una empresa puede tener muchas transacciones, y una tarjeta también puede estar asociada a múltiples transacciones.



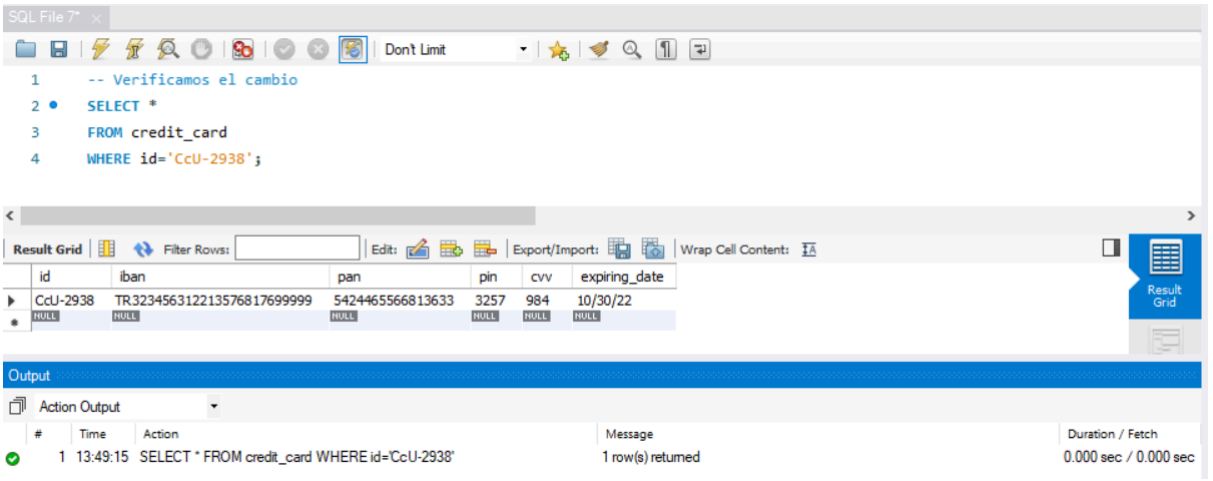
Ejercicio 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta asociado a su tarjeta de crédito con ID CcU-2938. La información que debe mostrarse para este registro es: TR323456312213576817699999. Recuerda mostrar que el cambio se realizó.

Utilizamos UPDATE para modificar la tabla “credit_card”, con el SET asignamos el nuevo valor al campo IBAN gracias a WHERE para una única fila con el id = 'CcU-2938', si se omite WHERE, se actualizan todas las filas.



Hacemos una consulta para demostrar todos los campos para el id = 'CcU-2938' que evidencia que el IBAN ahora es el correcto.

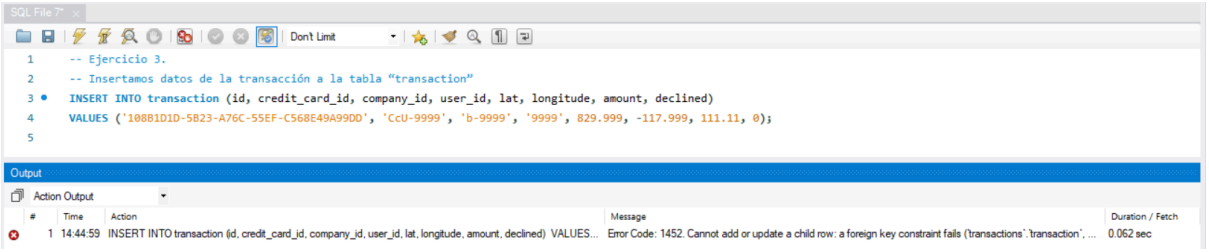


Ejercicio 3

En la tabla "transaction" ingresa una nueva transacción con la siguiente información:

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

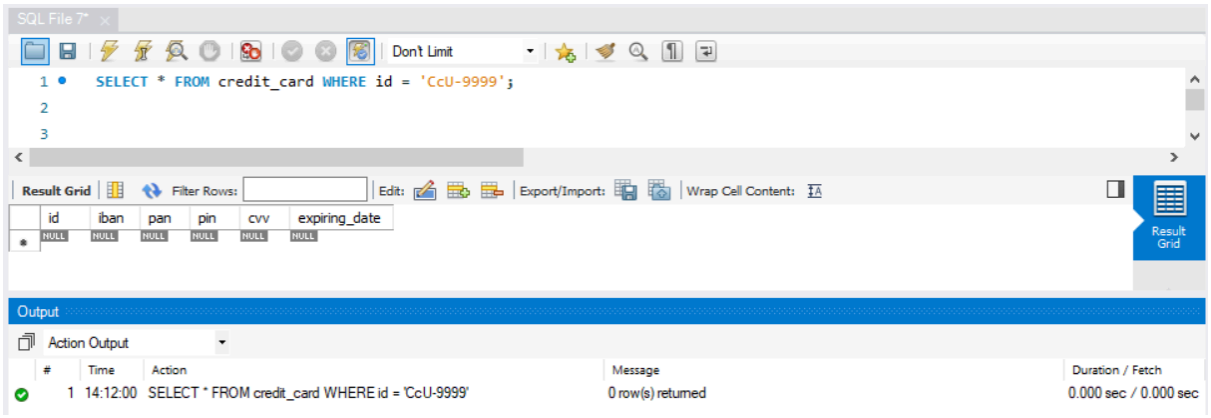
La transacción no se inserta porque estos valores violan restricciones de claves foráneas lo que afectaría la integridad referencial.



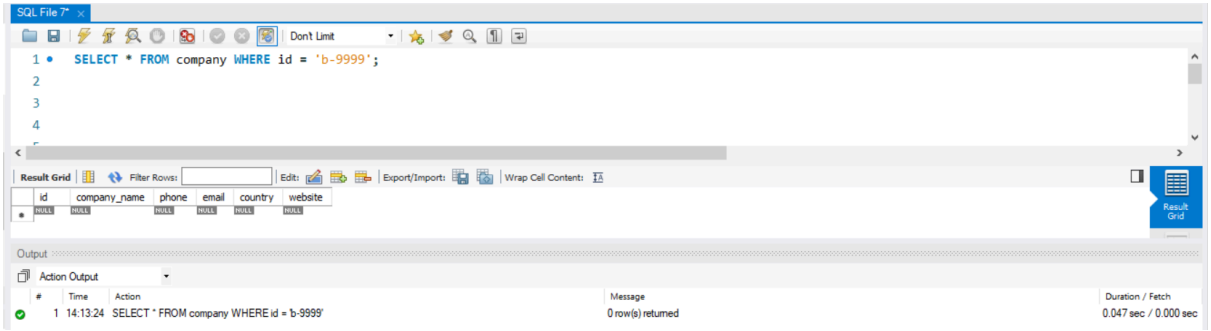
MySQL impide la inserción por los FOREIGN KEY de las columnas credit_card_id y company_id, que apuntan a las tablas credit_card y company respectivamente, así que solo permiten valores que existan en las tablas padre.

En concreto:

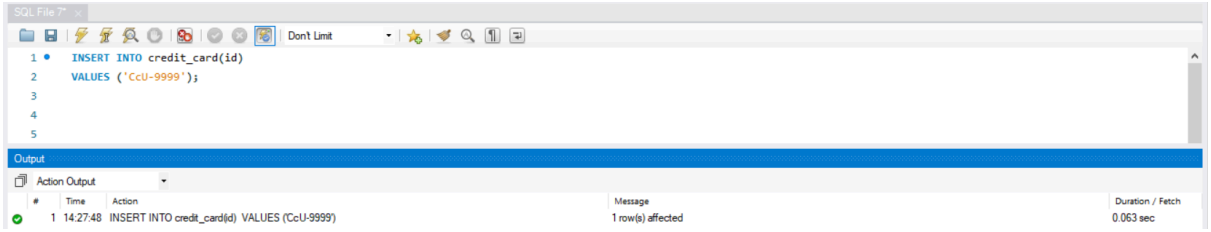
- credit_card_id = 'CcU-9999' no existe en la tabla “credit_card”:

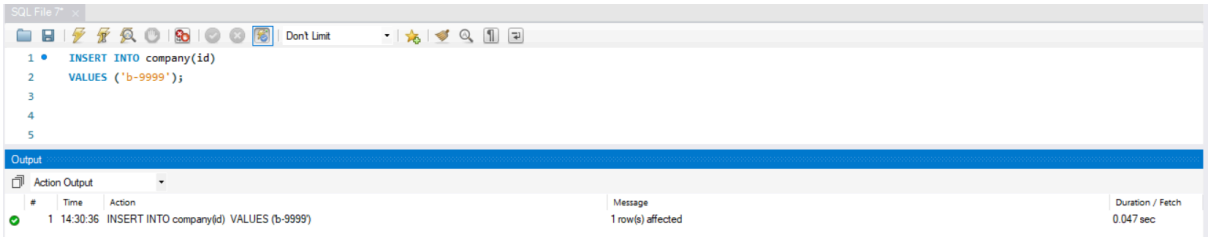


- company_id = 'b-9999' no existe en la tabla “company”:

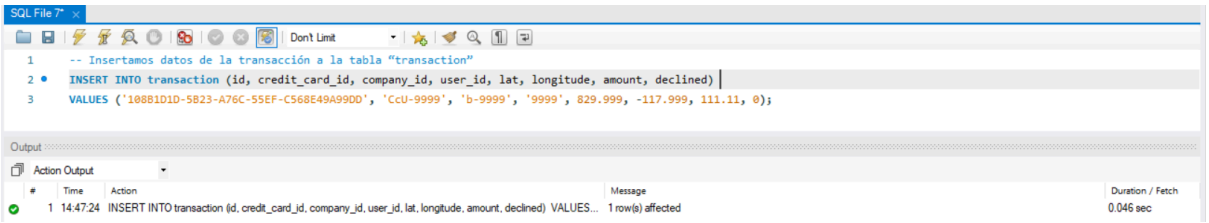


Para solucionarlo, primero se crean los registros padre en credit_card y company.

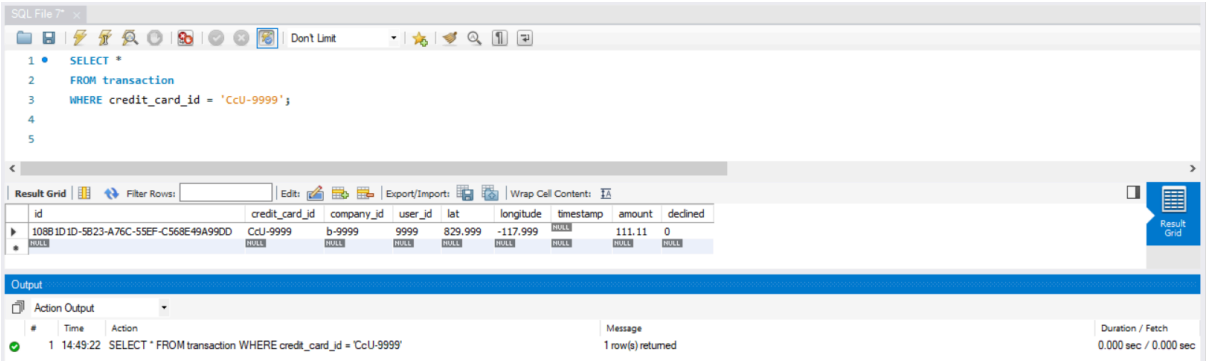




Después se vuelve a ejecutar la inserción de la nueva transacción, una vez añadidas las claves válidas, la transacción se inserta correctamente.



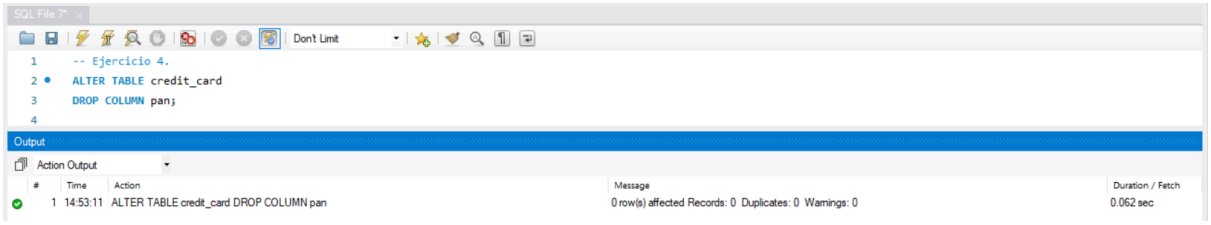
Lo comprobamos:



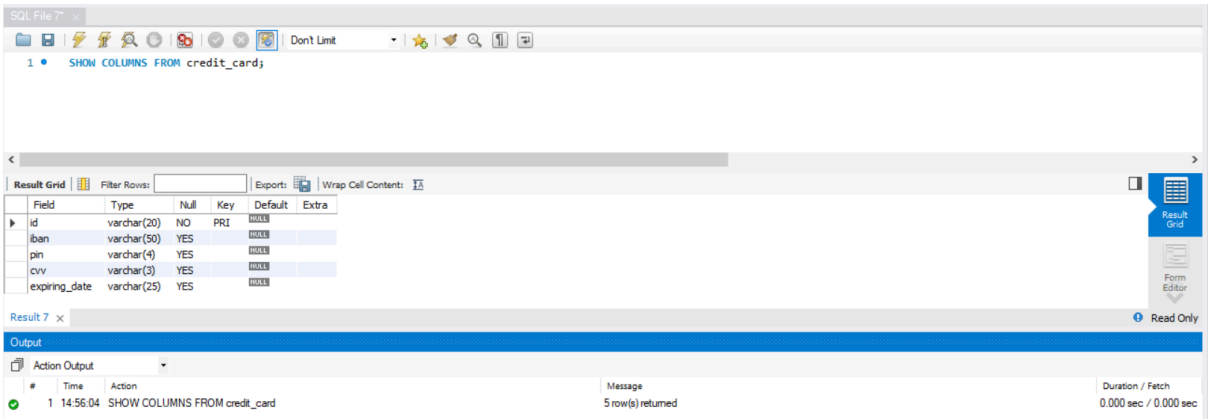
Ejercicio 4

Desde recursos humanos te solicitan eliminar la columna "pan" de la tabla credit_card. Recuerda mostrar el cambio realizado.

Modificamos la estructura de la tabla con ALTER TABLE, para eliminar la columna utilizamos DROP COLUMN.



Se verifica el cambio con la comanda SHOW COLUMNS FROM, donde ya no aparece dicha columna.

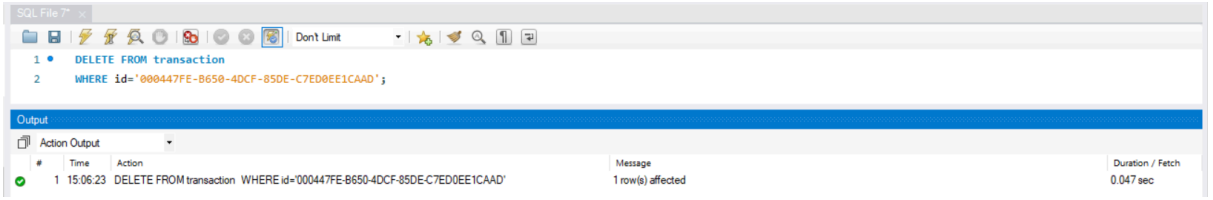


Nivel 2

Ejercicio 1

Elimina de la tabla transacción el registro con ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de datos.

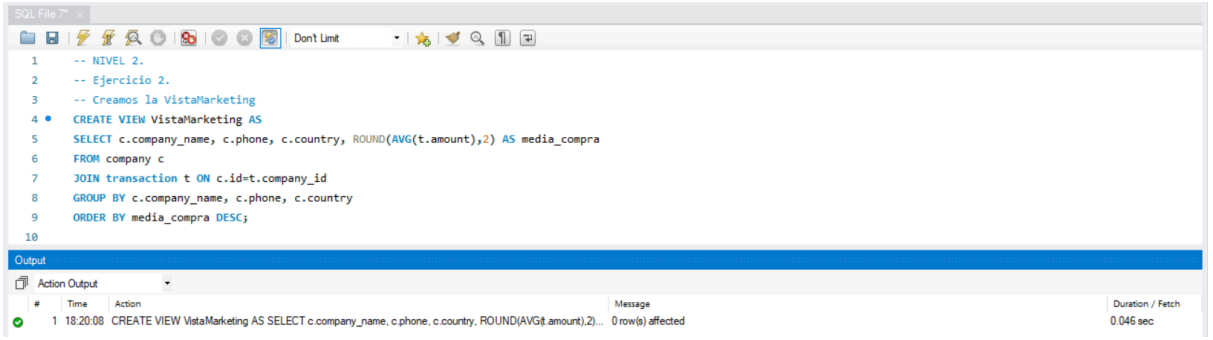
Para eliminar un registro se utiliza DELETE FROM indicando el nombre de la tabla y el criterio de la eliminación, en este caso el id de la transacción.



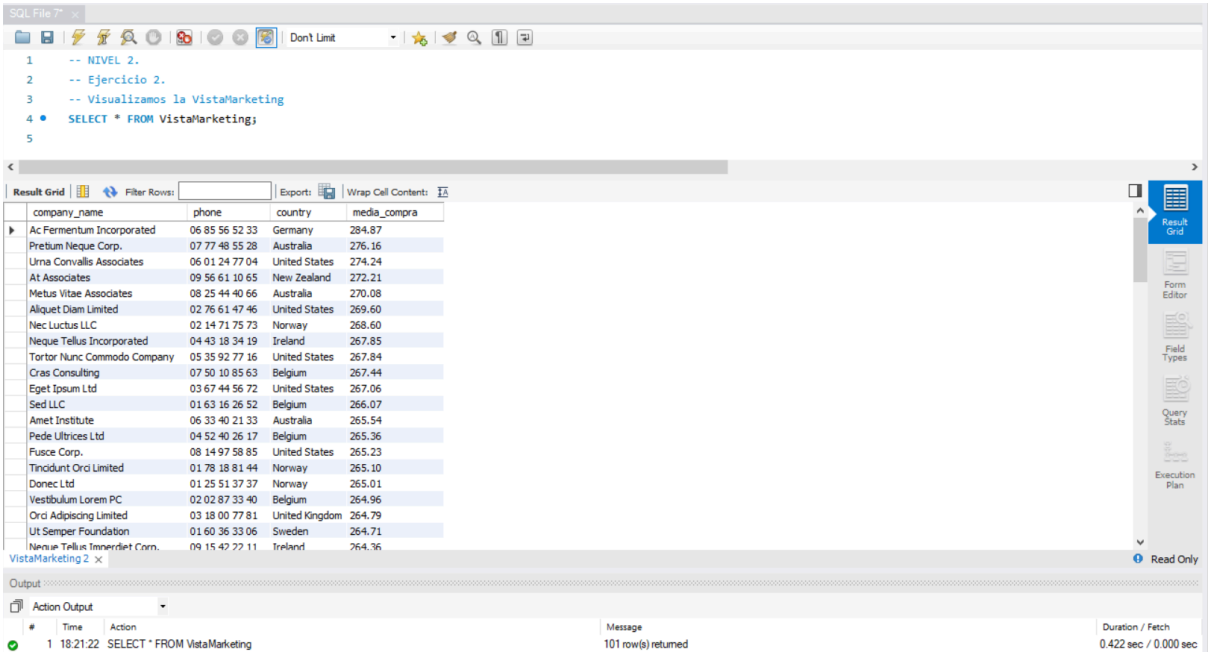
Ejercicio 2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesaria que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Media de compra realizado por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Creamos la vista VistaMarketing incluyendo las columnas solicitadas de las tablas “company” y “transaction”. Para relacionar ambas tablas utilizamos un JOIN. Es necesario agrupar por las columnas company_name, phone y country para poder calcular correctamente la media de compras (AVG(amount)), además, si no incluimos estas columnas en el GROUP BY, no podríamos mostrarlas en el SELECT y SQL daría error.



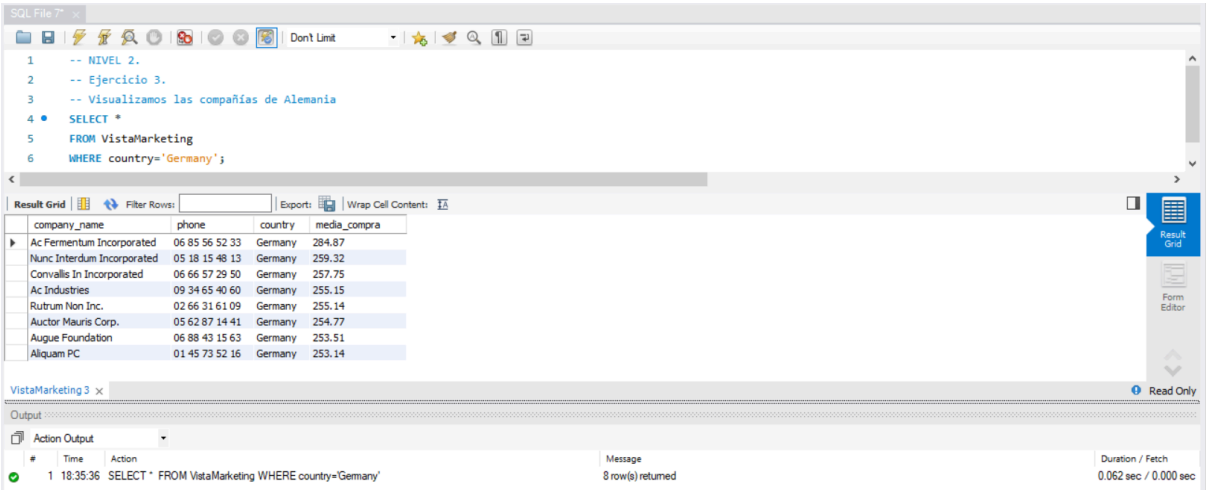
Para visualizar la información de la vista utilizamos un SELECT:



Ejercicio 3

Filtra la vista VistaMarketing para mostrar sólo las compañías que tienen su país de residencia en "Germany"

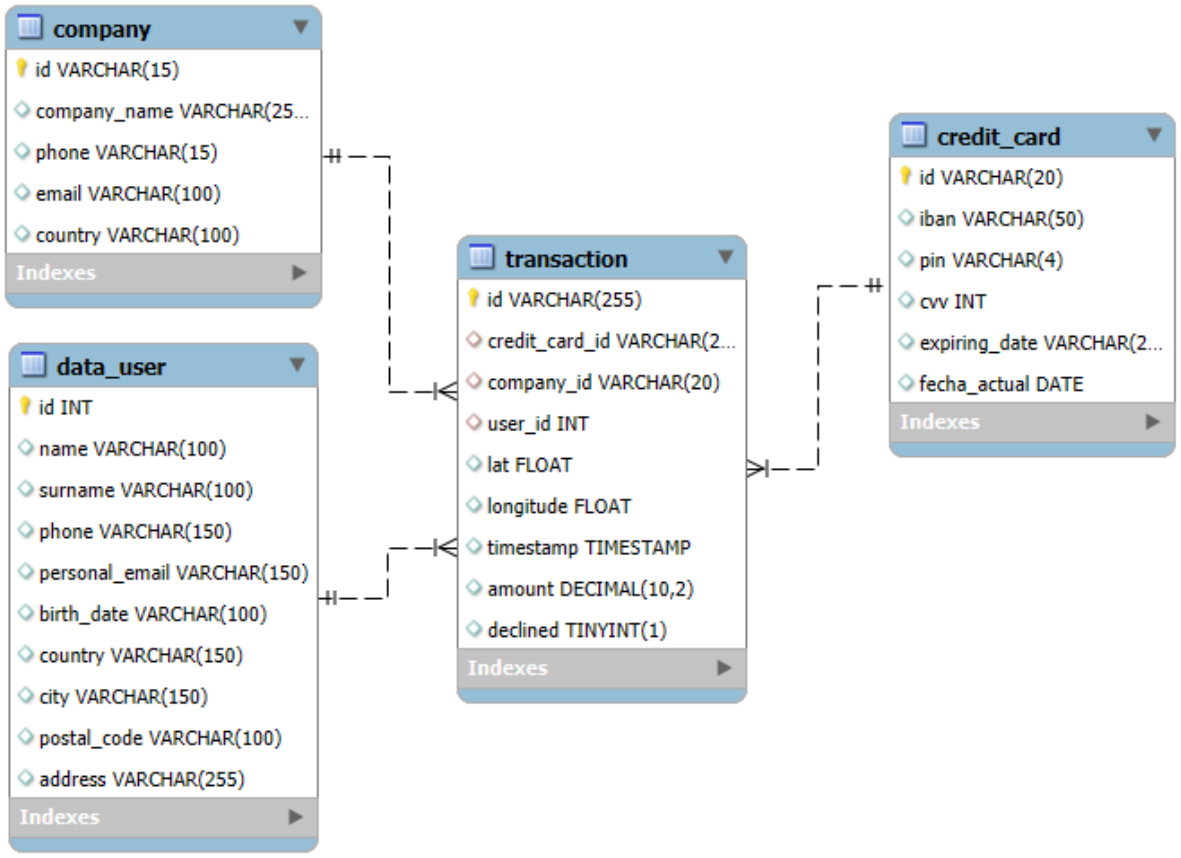
Utilizamos la VistaMarketing con WHERE para aplicar el filtro del país y únicamente elegir las compañías de Alemania.



Nivel 3

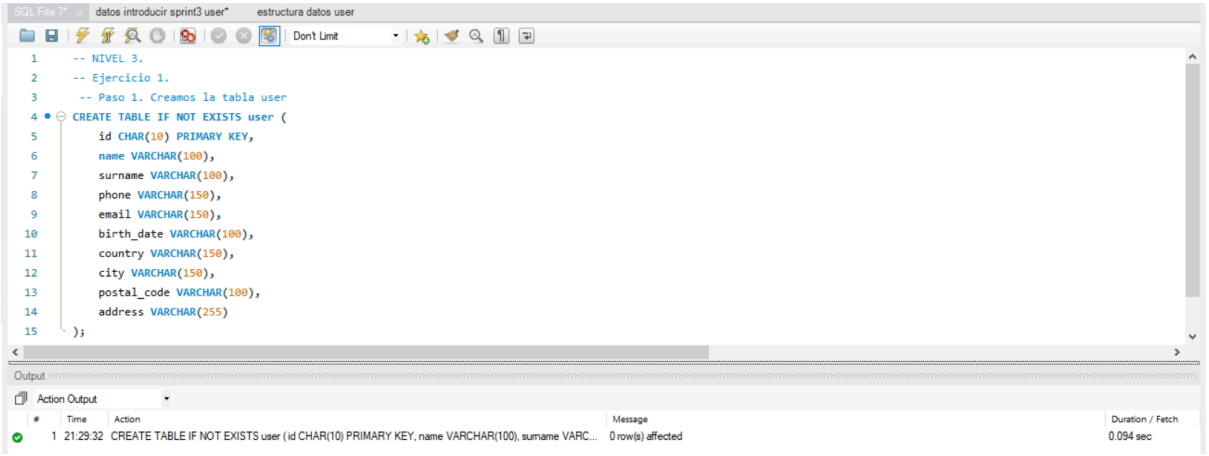
Ejercicio 1

La próxima semana tendrás una nueva reunión con los gerentes de marketing. Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Te pide que le ayudes a dejar los comandos ejecutados para obtener el siguiente diagrama:

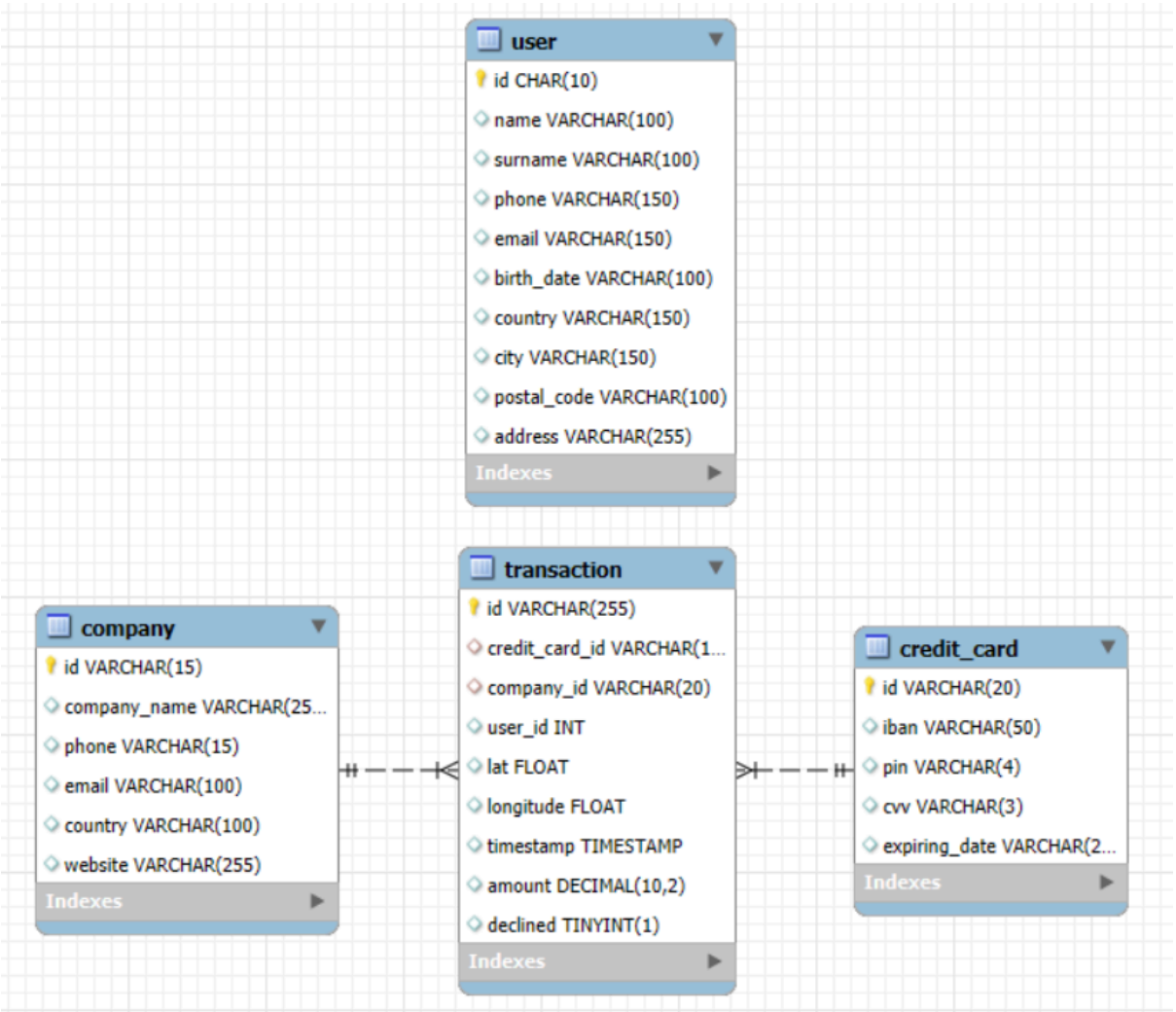


Para conseguir el esquema idéntico al presentado en la imagen hace falta realizar varios cambios. Crear la tabla “data_user” (a partir de la tabla “user”, posteriormente cambiando su nombre,tipo de “id” y nombre de columna “email”). Insertar los valores y conectarla con la tabla de hechos “transactions”. Además, realizar una serie de cambios en otras tablas: crear/eliminar columnas, cambiar el tipo de datos.

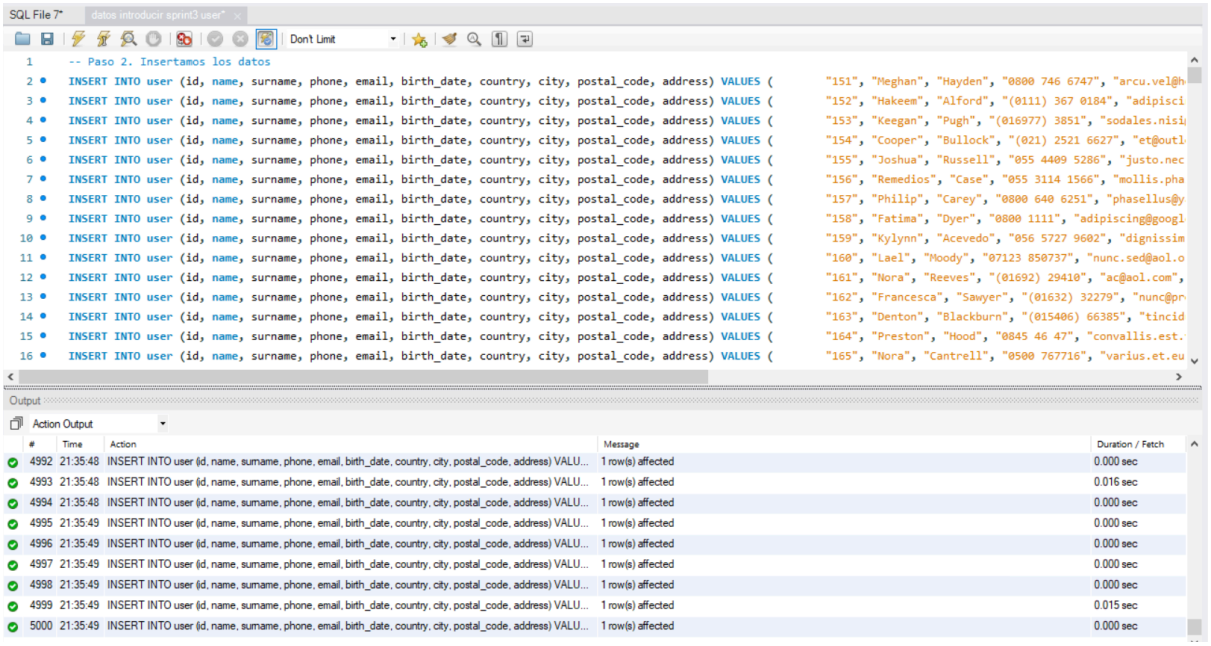
Paso 1. Creamos la tabla user a partir del archivo de la estructura:



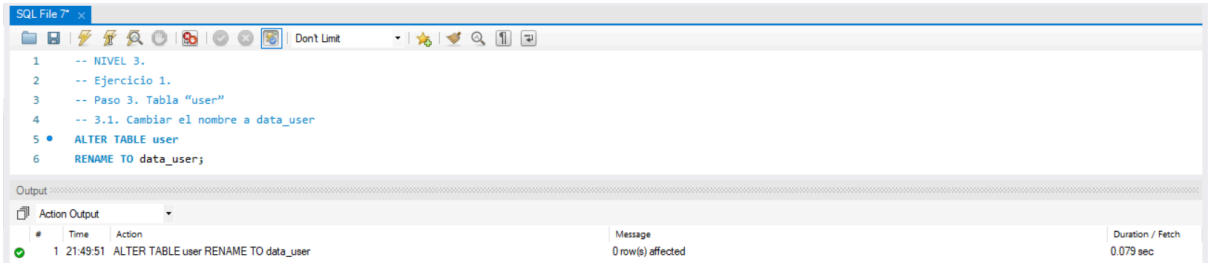
Como podemos observar se ha creado la tabla “user”, pero de momento no se relaciona con ninguna tabla. Hace falta crear la relación con “transaction” manualmente, pero antes de eso hay que verificar que los valores de user_id tengan correspondencia en data_user.id. MySQL no permite crear una foreign key si hay datos huérfanos, por lo que es necesario insertar los datos antes de establecer la relación.



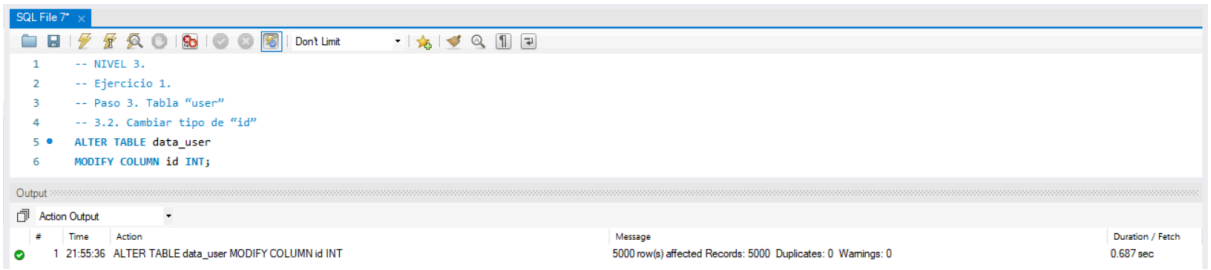
Paso 2. Insertamos los datos en la tabla



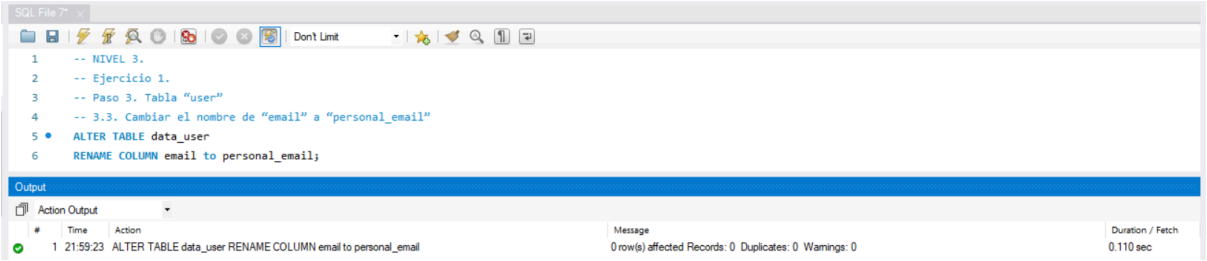
Paso 3. Tabla “user”: cambiar el nombre a data_user, cambiar tipo de “id”, cambiar el nombre de “email” a “personal_email”
3.1. Cambiar el nombre:



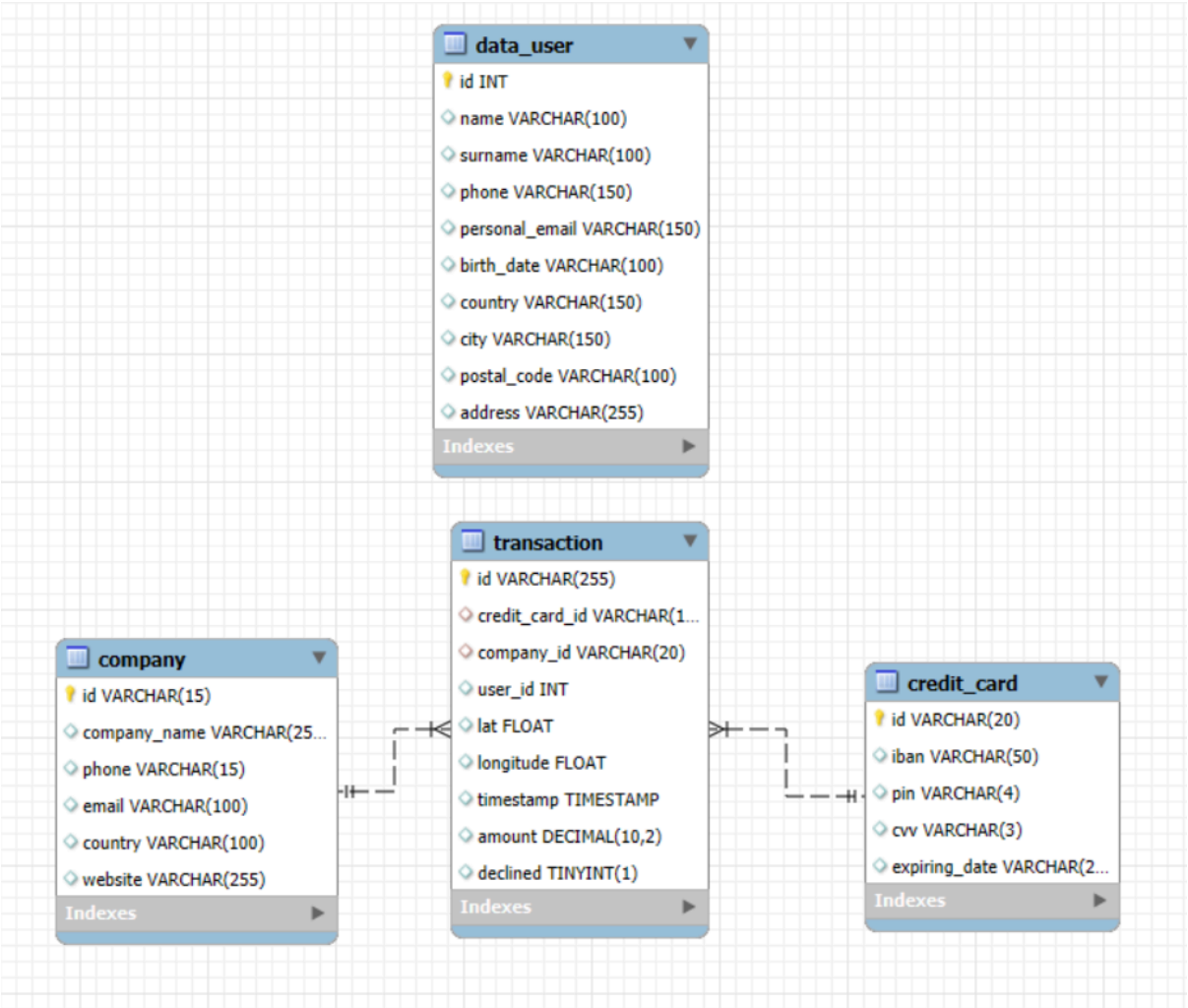
3.2. Cambiar tipo de “id”:



3.3. Cambiar el nombre de “email”

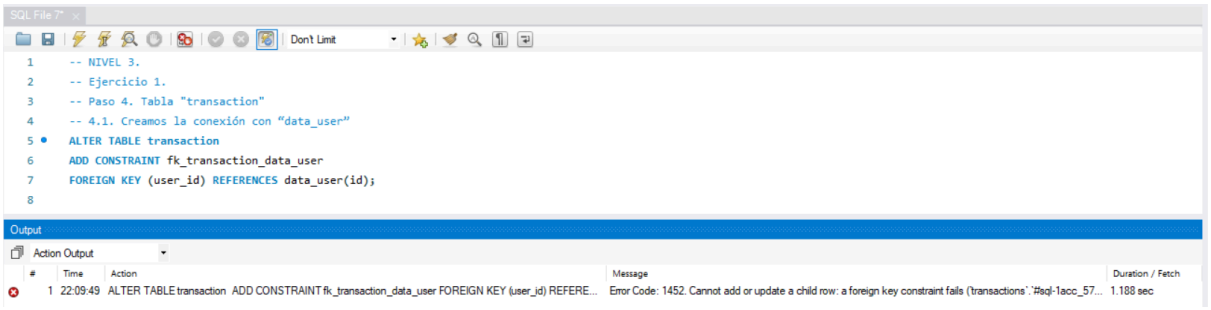


Observamos todos los cambios aplicados a la tabla data_user:

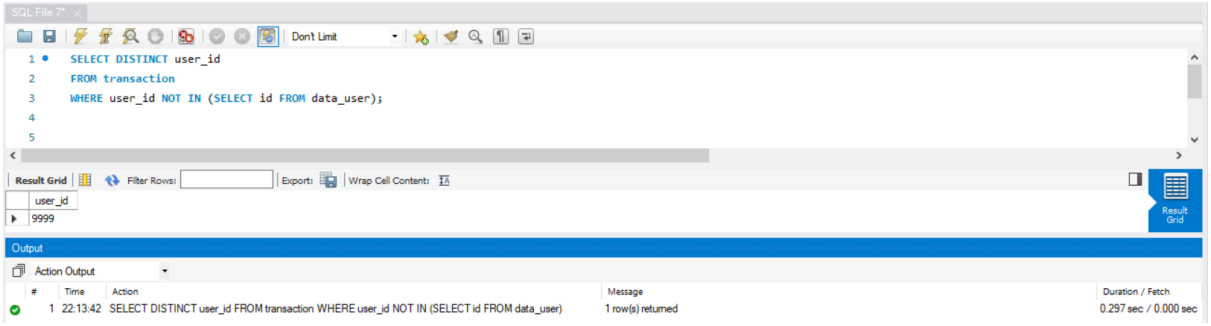


Paso 4. Tabla “transaction”: Creamos la conexión con “data_user” y cambiamos el número de carácteres de “credit_card_id”.

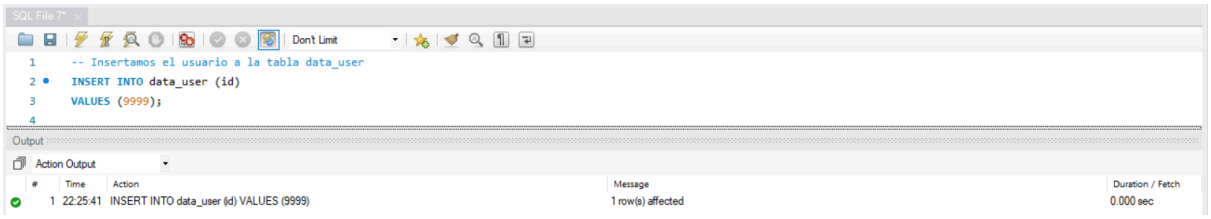
Al intentar crear la conexión aparece el error 1452, que significa que existen registros huérfanos:



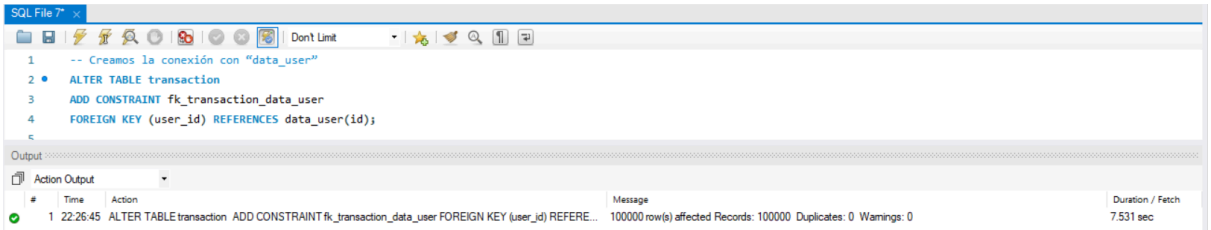
Revisamos los id “huérfanos”: los que tienen los datos en la tabla “transaction”, pero no en “data_user”, hemos encontrado un id “9999”.



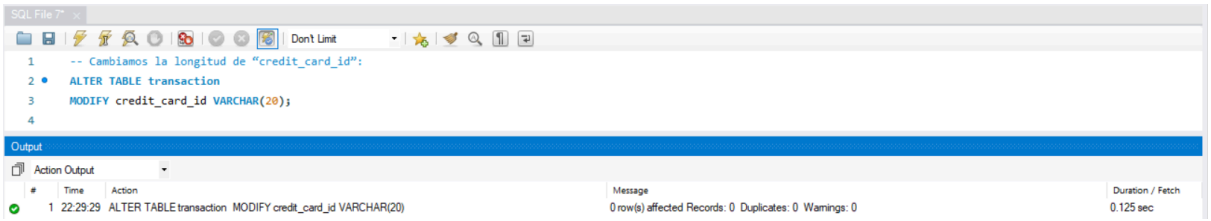
Para solucionar este problema hay que añadir el usuario con el id “9999” a la tabla data_user.



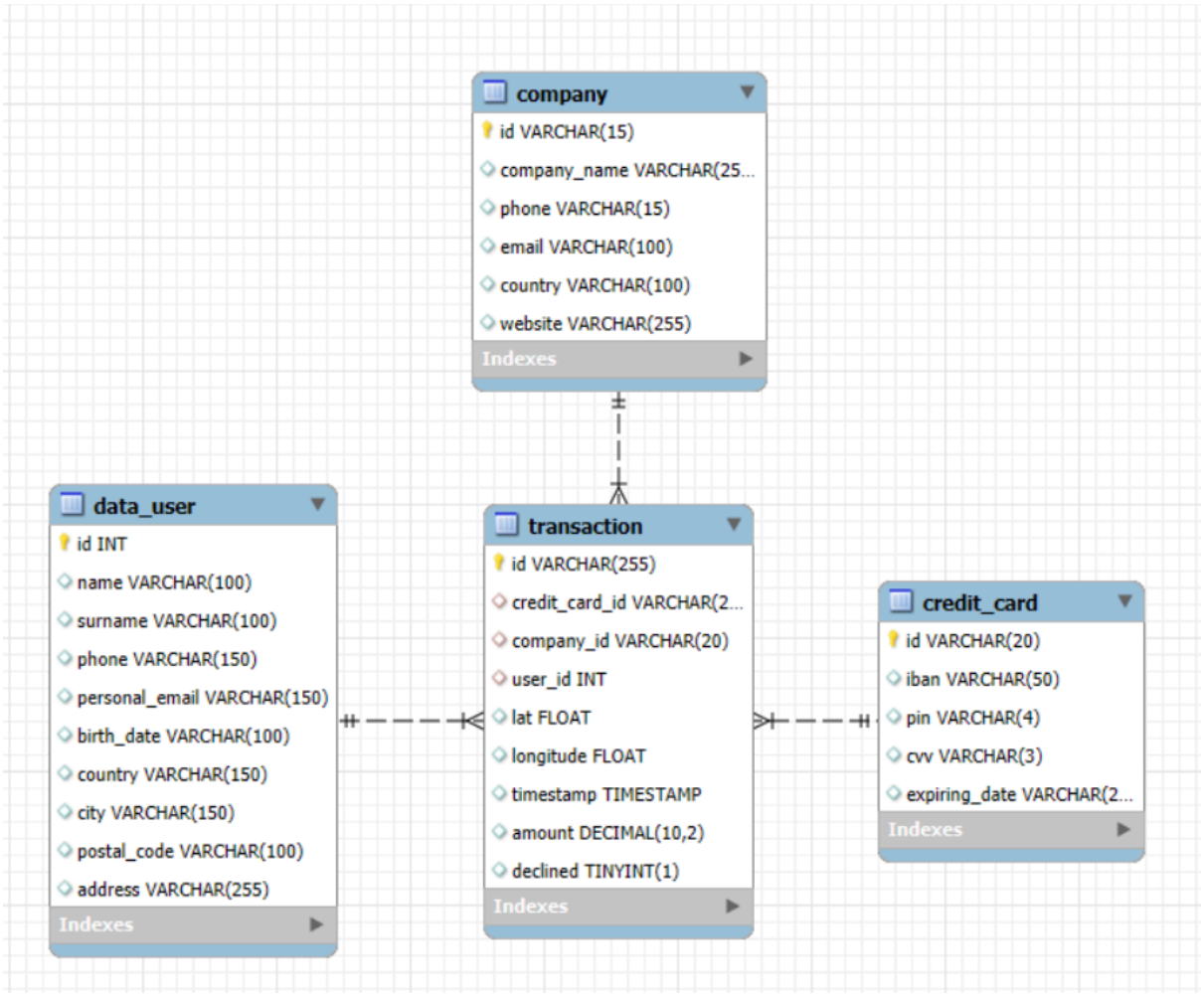
Creamos la conexión con “transaction”:



Cambiamos la longitud de “credit_card_id”:

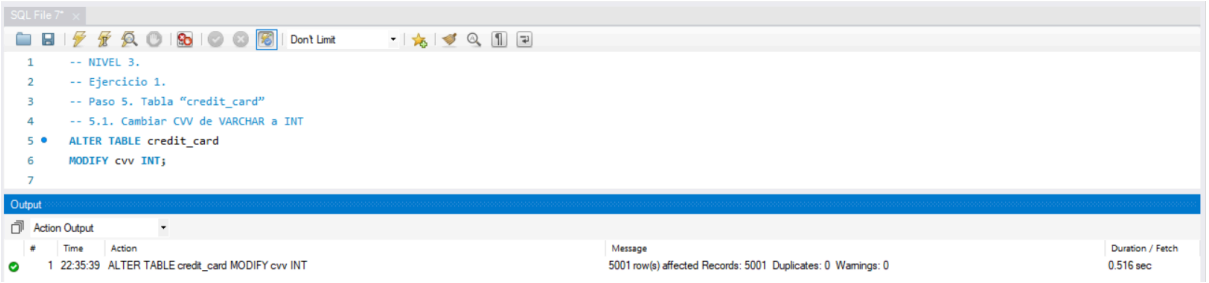


Observamos todos los cambios aplicados a la tabla “transaction” y al esquema de la BBDD “transactions”:

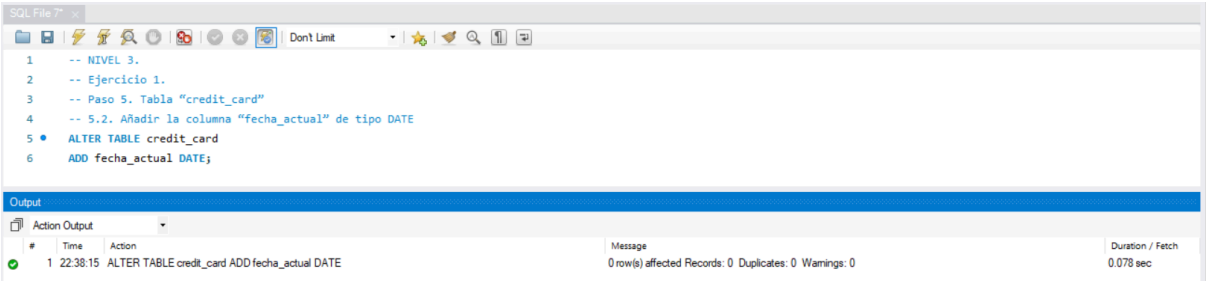


Paso 5. Seguimos con la tabla “credit_card”: cambiar CVV de VARCHAR a INT y añadir la columna “fecha_actual” de tipo DATE

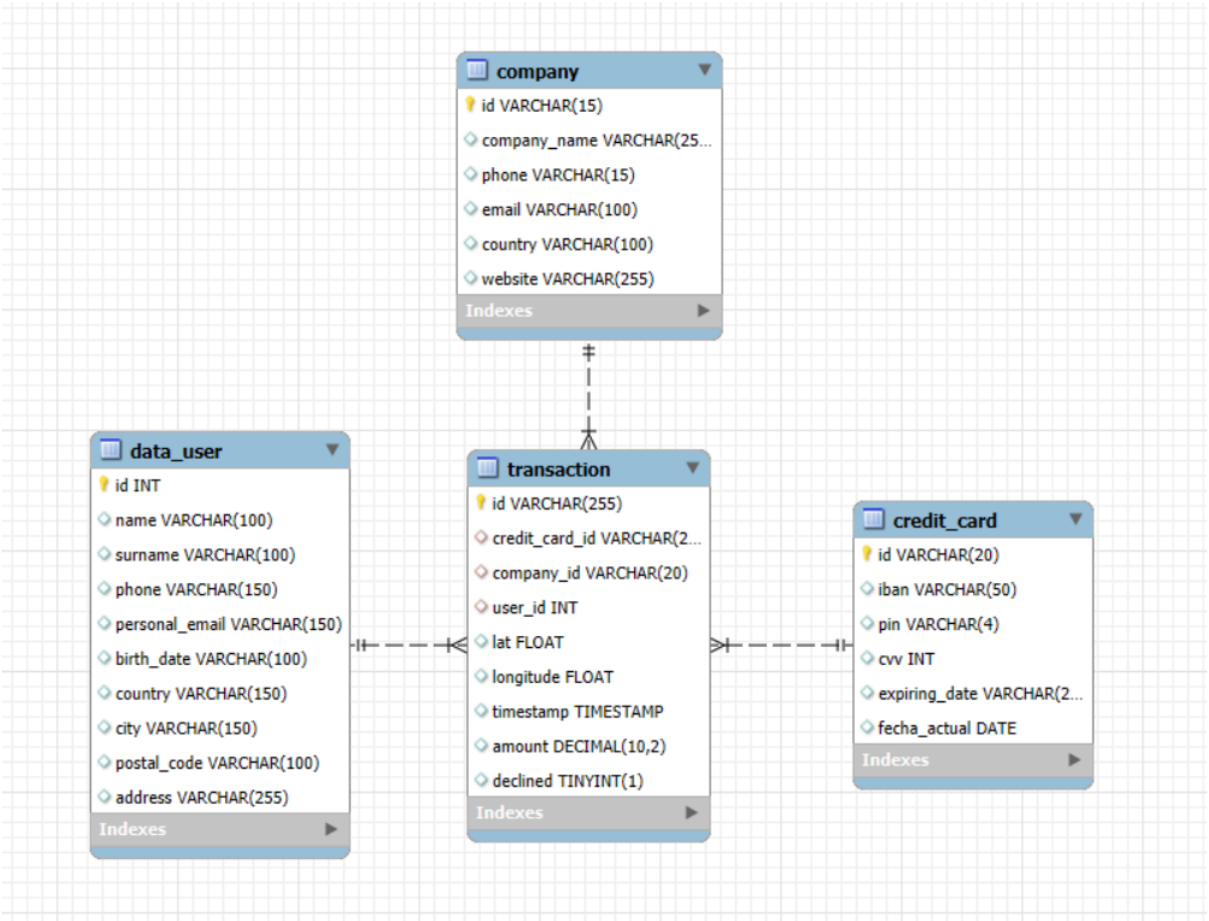
Cambiar el tipo de CVV:



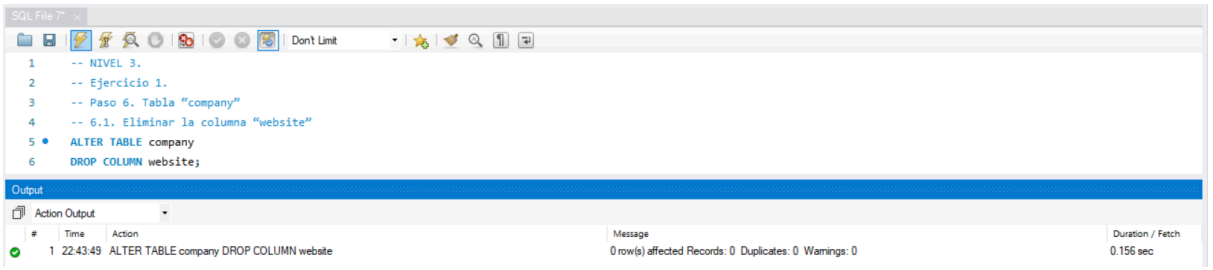
Añadir la columna “fecha_actual”:



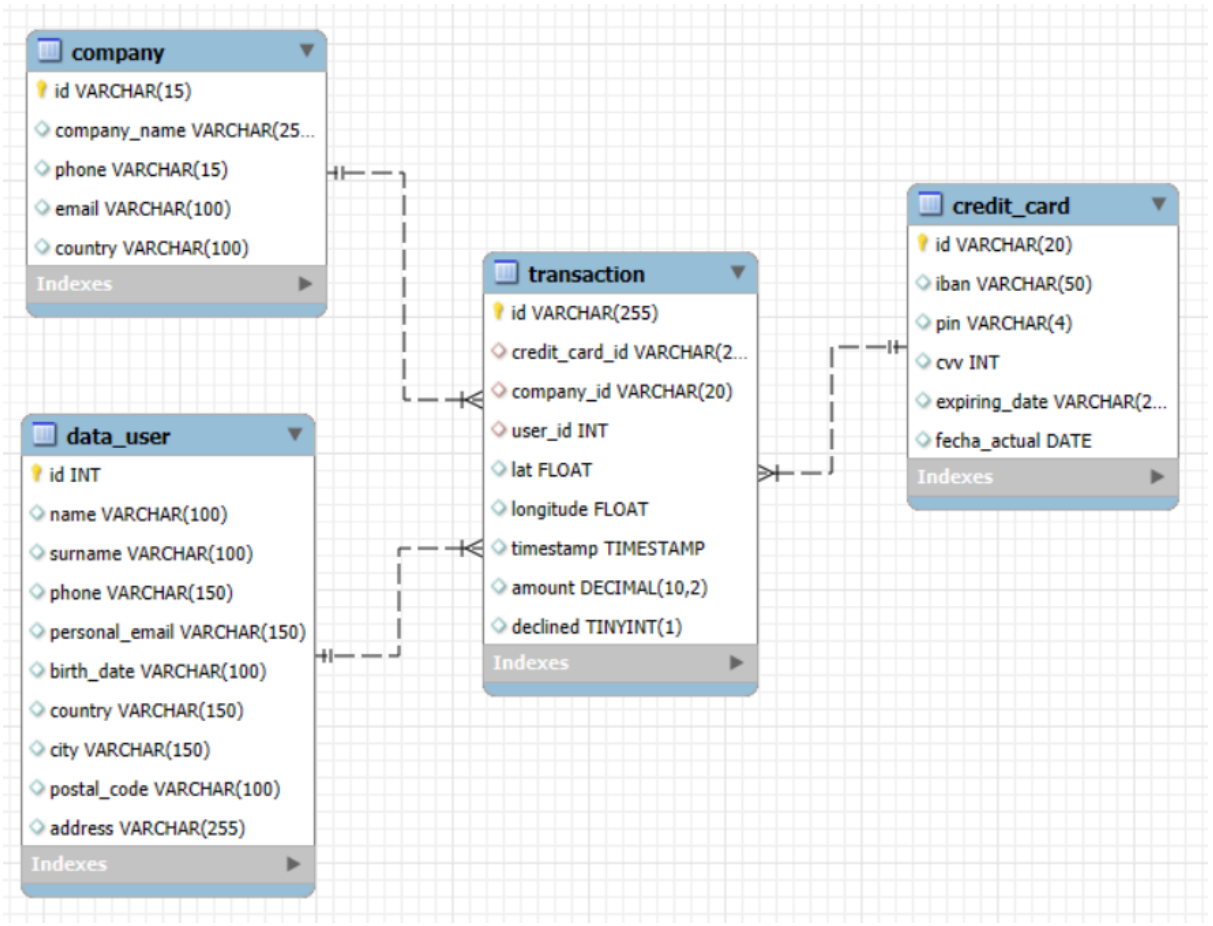
Observamos en el esquema los cambios realizados en la tabla “credit_card”:



Paso 6. Continuamos con la tabla “company” y eliminamos la columna “website”.



Comprobamos si el esquema actual después de todos los cambios coincide con el esquema deseado:



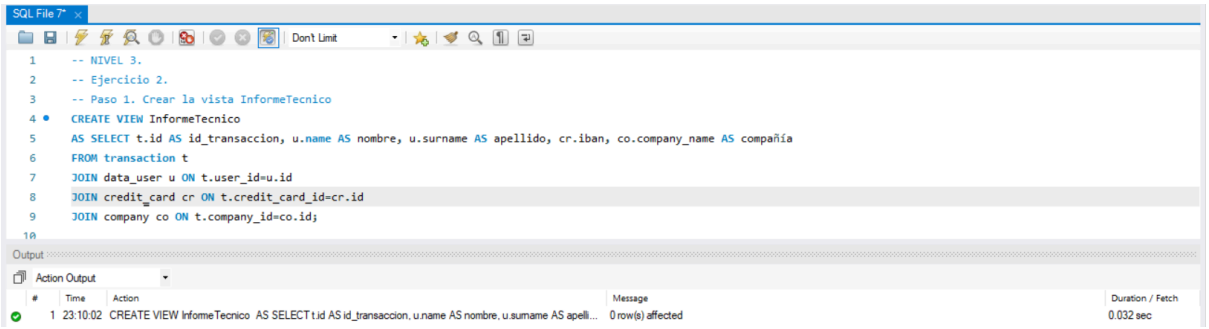
Ejercicio 2

La empresa también le pide crear una vista llamada "InformeTecnico" que contenga la siguiente información:

- ID de la transacción
- Nombre del usuario/a
- Apellido del usuario/a
- IBAN de la tarjeta de crédito usada.
- Nombre de la compañía de la transacción realizada.
- Asegúrese de incluir información relevante de las tablas que conocerá y utilice alias para cambiar de nombre columnas según sea necesario.

Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción.

Creo la vista InformeTecnico combinando las tablas transaction, data_user, credit_card y company mediante JOIN usando sus claves foráneas.
Selecciono únicamente las columnas relevantes y utilizo alias para hacer los nombres más claros.



Dado que MySQL no permite usar ORDER BY dentro de la vista, aplico la ordenación cuando la consulto: