DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

SPRINT 03

Manipulación de tablas
30 de abril de 2025

Letiane Benincá
benincalf@gmail.com

IT ACADEMY
Barcelona Activa

DESCRIPCIÓN

En este sprint, se simula una situación empresarial en la que debes realizar varias manipulaciones en las tablas de la base de datos. A su vez, tendrás que trabajar con índices y vistas. En esta actividad, continuarás trabajando con la base de datos que contiene información de una empresa dedicada a la venta de productos en línea. En esta tarea, comenzarás a trabajar con información relacionada con tarjetas de crédito.

NIVEL 1

EJERCICIO 01

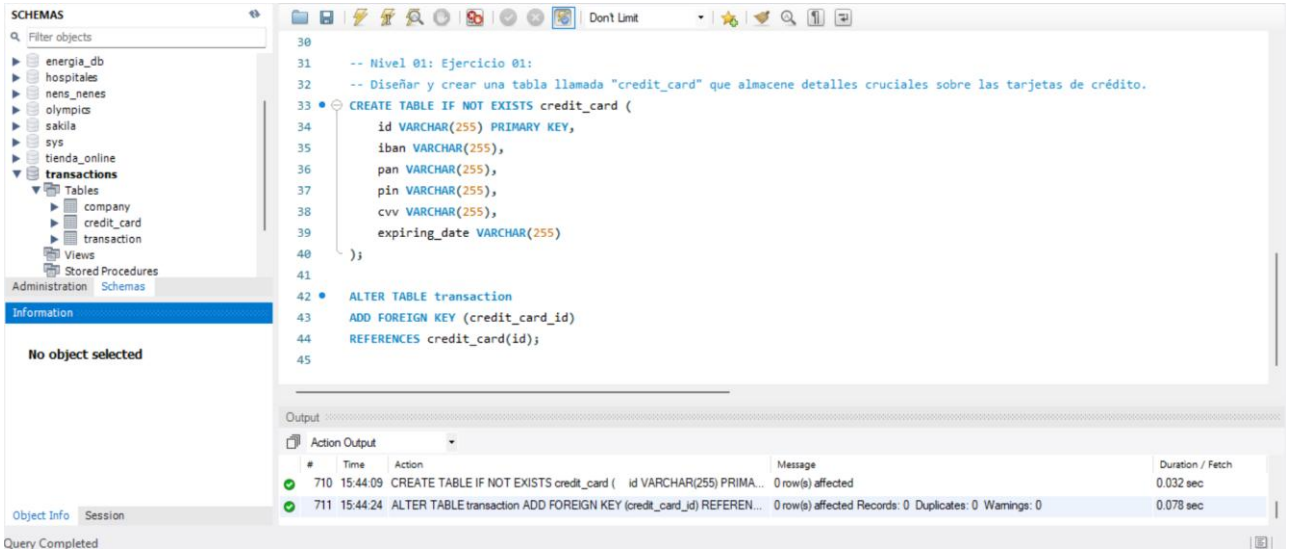
Tu tarea es diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito. La nueva tabla debe ser capaz de identificar de manera única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "company"). Después de crear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

Creación de la tabla credit_card:
La tabla fue creada de acuerdo con la lógica de los datos a introducir de crédito. Los campos creados fueron:

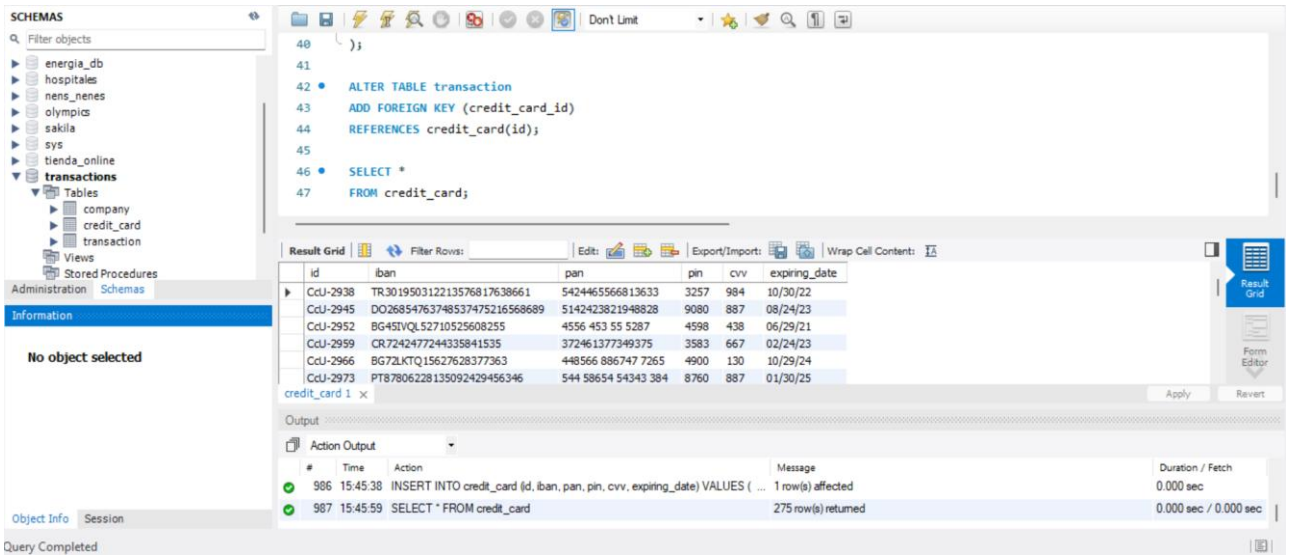
- **id VARCHAR(255):** clave primaria, alfanuméricos de hasta 15 caracteres. [‘CcU-2938’]
- **Iban VARCHAR(255):** International Bank Account Number, alfanuméricos de hasta 50 caracteres. ['TR301950312213576817638661]
- **pan VARCHAR(255):** Primary Account Number, almacena el número de la tarjeta, alfanuméricos de hasta 20 caracteres. ['5424465566813633']
- **pin VARCHAR(255):** Código secreto de la tarjeta, alfanuméricos de hasta 4 caracteres. ['3257']
- **cvv VARCHAR(255):** Card Verification Value, alfanuméricos de hasta 3 caracteres. ['984']
- **expiring_date VARCHAR(255):** la fecha de caducidad de la tarjeta en formato alfanuméricos de hasta 10 caracteres. ['10/30/22'] # He intentado utilizar el formato DATE, pero el default de MySQL es XX/XX/XXXX, lo que generaba un error a utilizarlo.

**PS: Siempre iniciar con VARCHAR(255), cuando no conocemos los datos para evitar errores de carga. Luego si puede adaptar.*

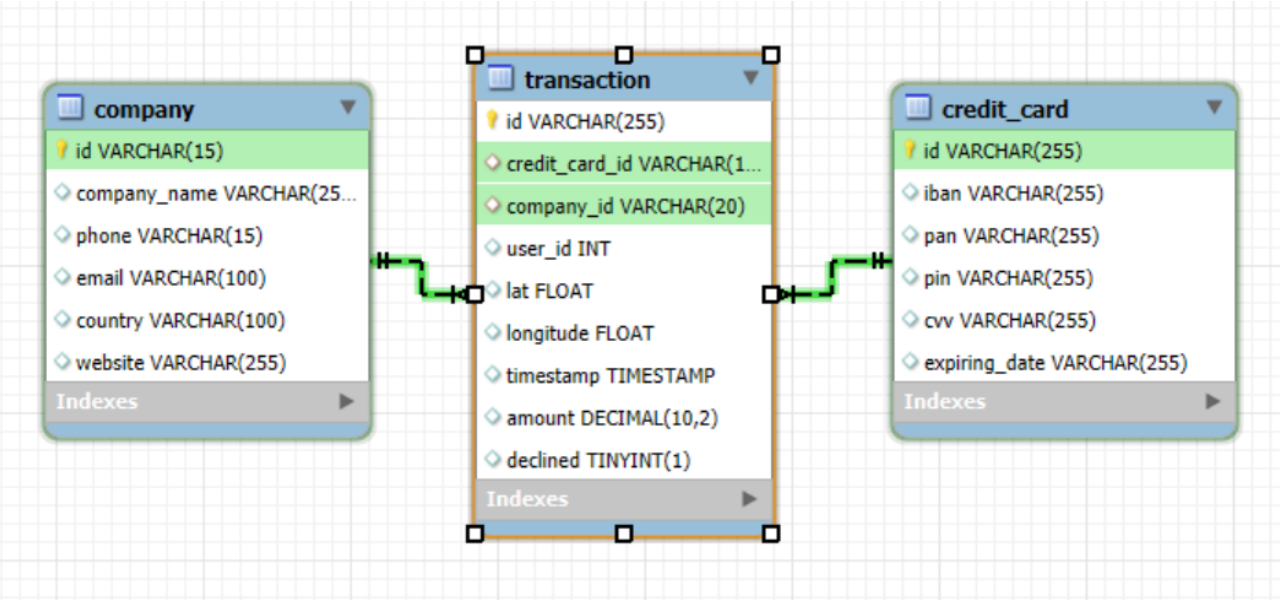
Añadí una clave foránea en la tabla "transaction" que vincula el campo "credit_card_id" con el "id" de la tabla "credit_card".



Finalmente, inserté los datos del archivo "datos_introducir_credit".



Utilizando el reverse engineer, verificamos las cardinalidades de las tablas. Inicialmente la clave foránea de la nueva tabla no estaba bien configurada y para esto si ha añadido una restricción para la misma. Presentamos el diagrama final:

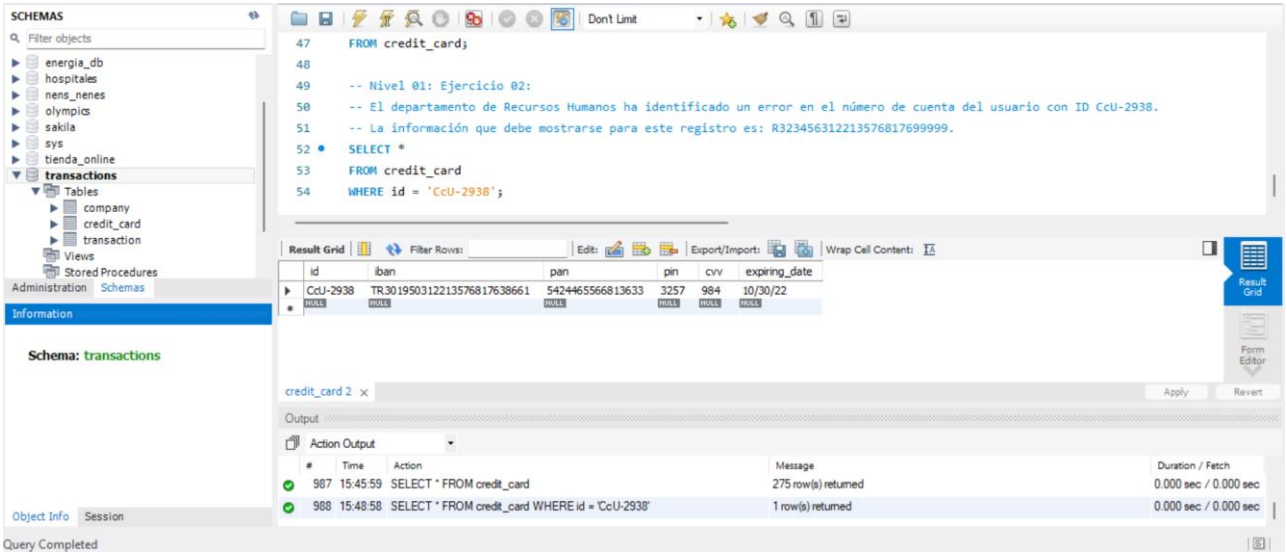


La base de datos es relacional con un esquema en estrella, donde la tabla **transaction** es la tabla de hechos y **company** y **credit_card** son tablas dimensionales. Las relaciones entre las tablas son de tipo 1 a N, ya que una empresa o tarjeta de crédito pueden estar asociadas a múltiples transacciones, pero cada transacción es única. Las claves primarias (PK) de las tablas dimensionales (id) se vinculan como claves foráneas (FK) en la tabla de hechos.

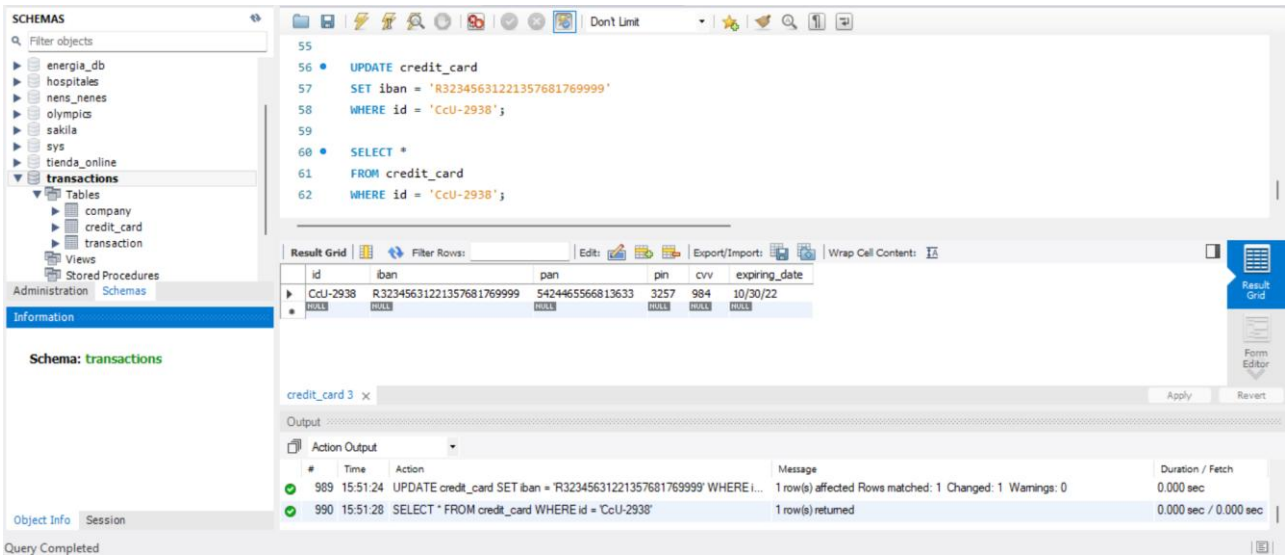
EJERCICIO 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. La información que debe mostrarse para este registro es: R323456312213576817699999. Recuerda demostrar que el cambio se ha realizado.

En este caso, primero conferimos los datos existentes:



Por medio del uso del UPDATE TABLE, hacemos el cambio de información del usuario ‘CcU-2938’.

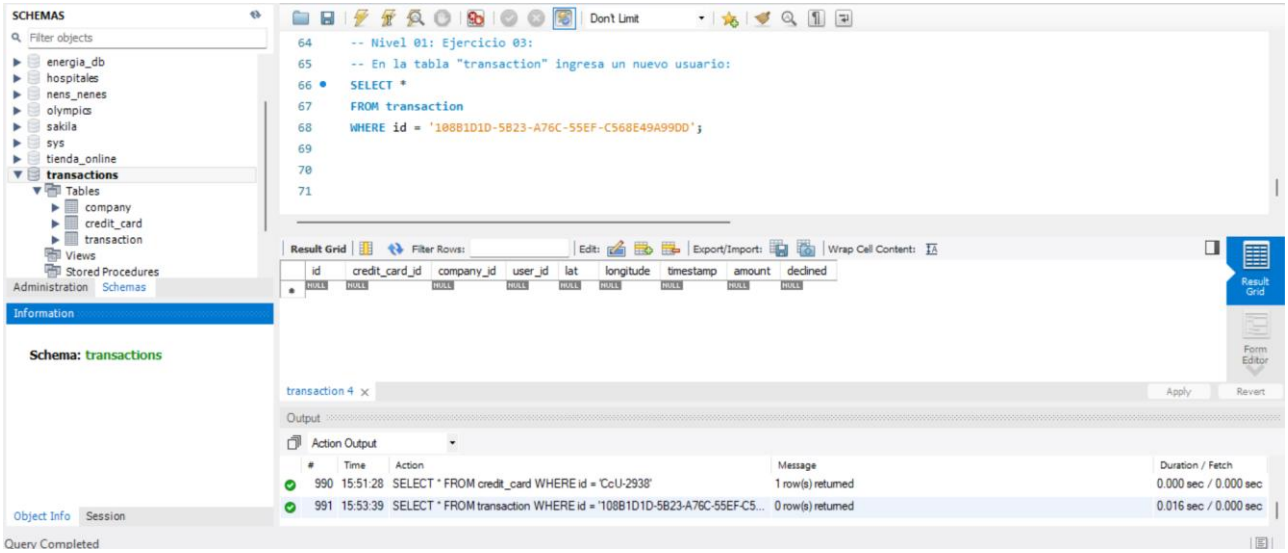


EJERCICIO 3

En la tabla "transaction" ingresa un nuevo usuario con la siguiente información:

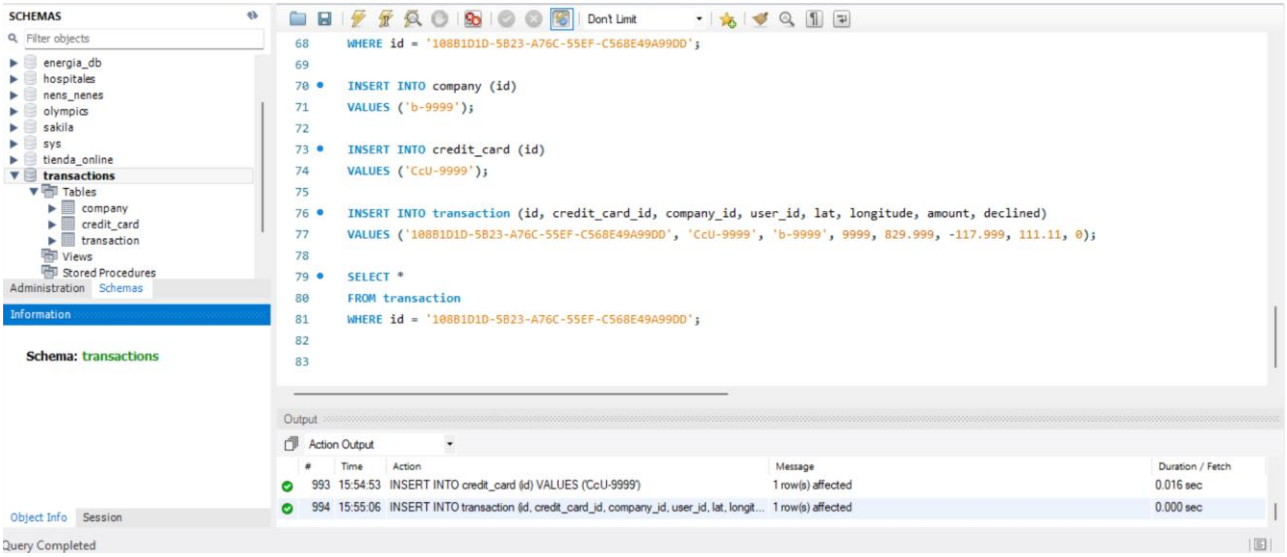
Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

Inicialmente, verificamos se no existen los datos en la tabla:

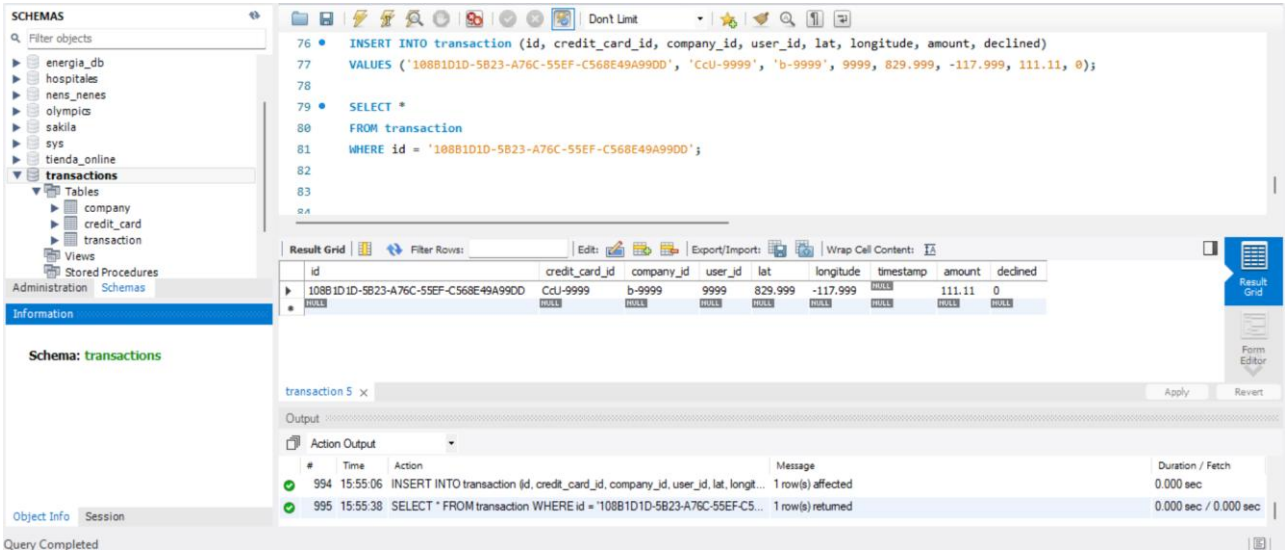


Para añadir los datos de nuevo usuario fue utilizado el INSER INTO:

- Inicialmente en las tablas **company** y **credit_card**, luego inserimos los datos en la tabla **transaction**, quedando fuera el timestamp que no estaba en los datos, siendo que este quedará NULL.



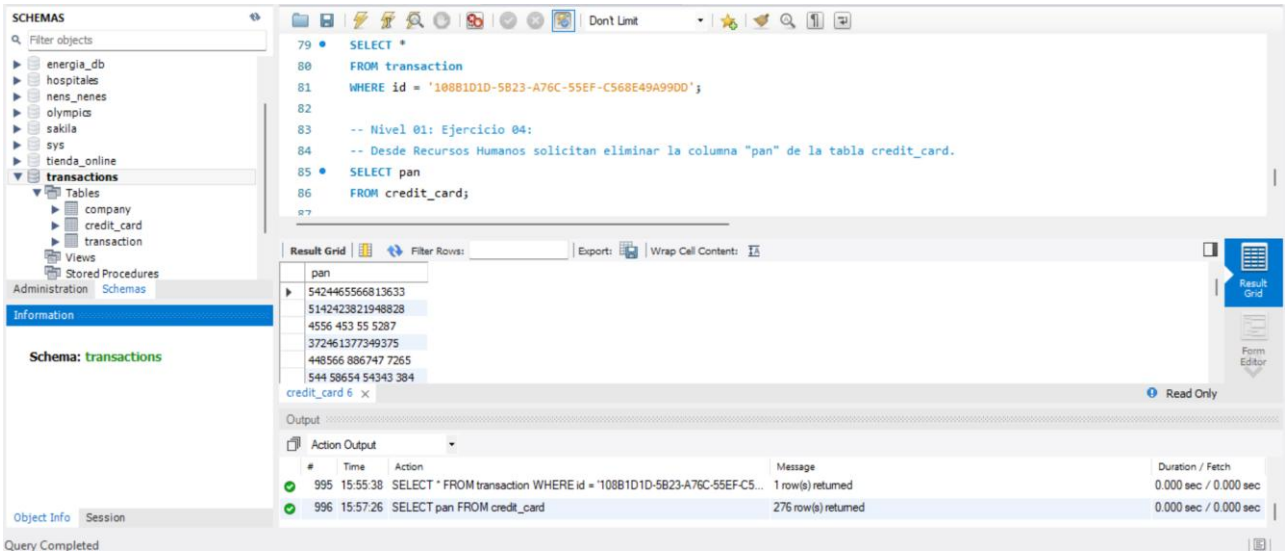
Verificamos la inserción de los datos:



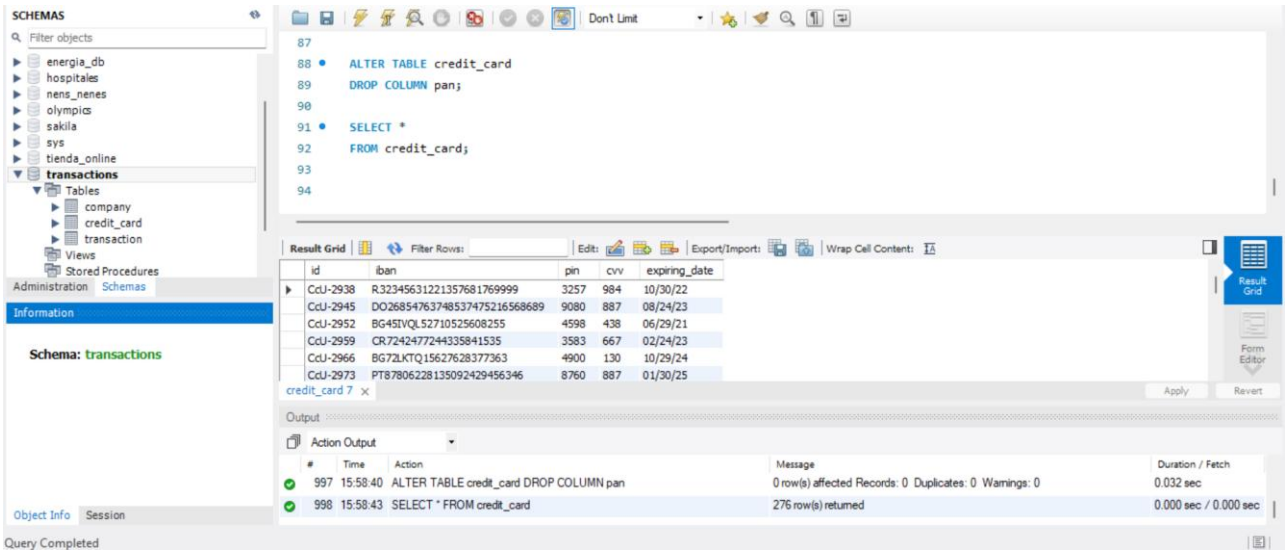
EJERCICIO 4

Desde Recursos Humanos solicitan eliminar la columna "pan" de la tabla *credit_card*. Recuerda demostrar que el cambio se ha realizado.

Verificamos la tabla *credit_card* la coluna pan:



Para alterar la tabla, utilizamos el ALTER TABLE credit_card DROP COLUMN pan eliminamos la columna. Así queda el resultado, la tabla credit_card ya no tiene la columna pan.

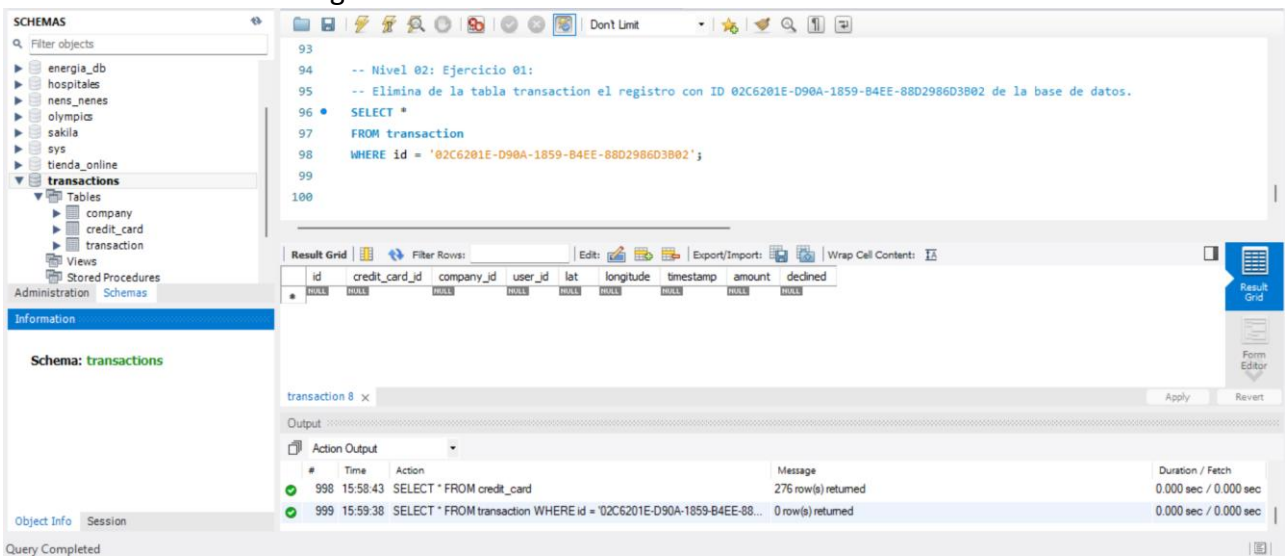

SCHEMAS
Filter objects
energia_db
hospitales
nens_nenes
olympics
sakila
sys
tienda_online
transactions
Tables
company
credit_card
transaction
Views
Stored Procedures
Administration
Schemas
Information
Schema: transactions
Object Info
Session
Don't Limit
87
88 ALTER TABLE credit_card
89 DROP COLUMN pan;
90
91 SELECT *
92 FROM credit_card;
93
94
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
id
iban
pin
cvv
expiring_date
CcU-2938 R323456312213576817699999 3257 984 10/30/22
CcU-2945 DO2685476374853747521656868 9 9080 887 08/24/23
CcU-2952 BG45IVQL52710525608255 4598 438 06/29/21
CcU-2959 CR7242477244335841535 3583 667 02/24/23
CcU-2966 BG72LKTQ15627628377363 4900 130 10/29/24
CcU-2973 PT8780622813509242945 6346 8760 887 01/30/25
credit_card 7
Apply
Revert
Result Grid
Form Editor
Output
Action Output
Time Action Message Duration / Fetch
997 15:58:40 ALTER TABLE credit_card DROP COLUMN pan 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.032 sec
998 15:58:43 SELECT * FROM credit_card 276 row(s) returned 0.000 sec / 0.000 sec
Query Completed

NIVEL 2

EJERCICIO 01

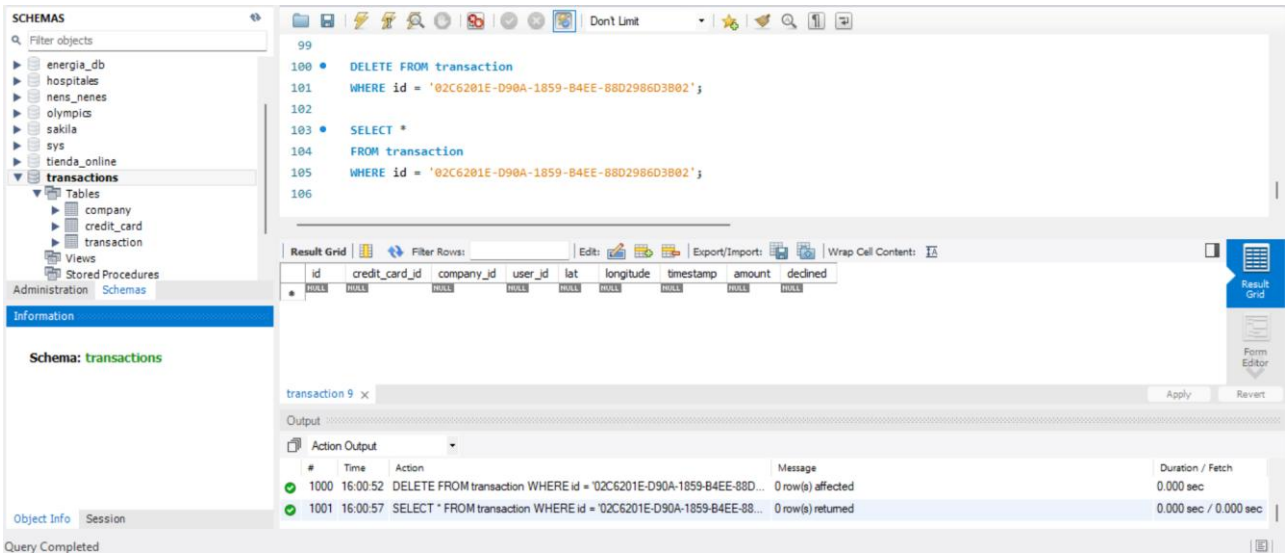
Elimina de la tabla *transaction* el registro con ID *02C6201E-D90A-1859-B4EE-88D2986D3B02* de la base de datos.

Primero verificamos el registro:



Para eliminarlo utilizamos el DELETE FROM con el filtro WHERE con el código del registro:

Una vez eliminado, volvemos a seleccionar para comprobar que se haya eliminado correctamente.



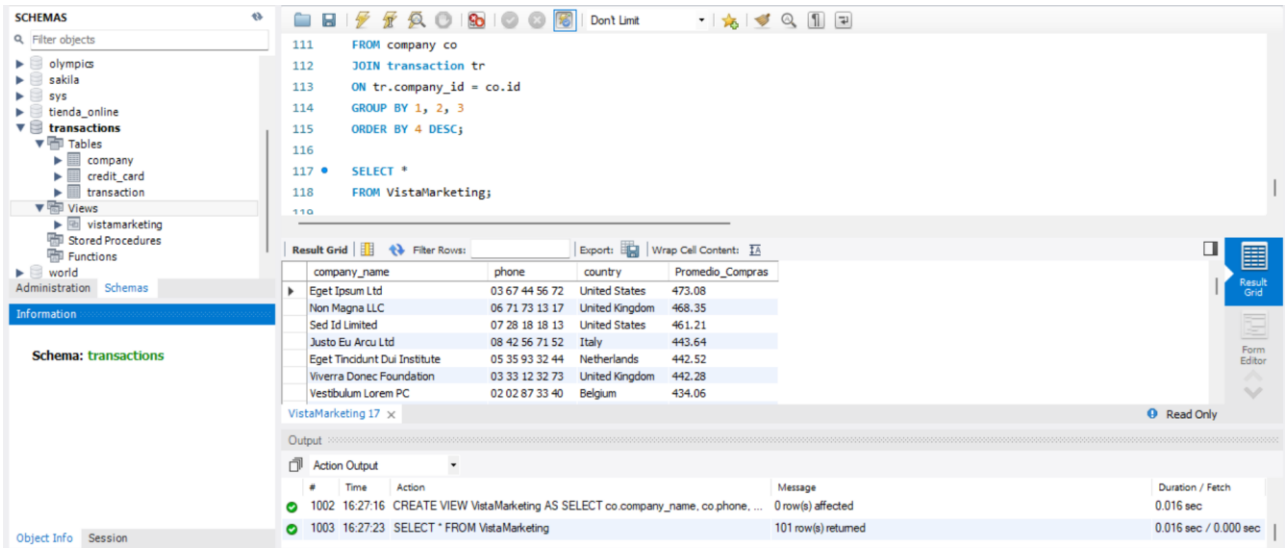
EJERCICIO 2

El departamento 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 necesario que crees una vista llamada *VistaMarketing* que contenga la siguiente información:

- Nombre de la compañía.
- Teléfono de contacto.
- País de residencia.
- Promedio de compra realizada por cada compañía.

Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Usamos el comando **CREATE VIEW AS** para crear la vista. Seleccionamos el nombre de la compañía, el teléfono y el país, además de calcular el promedio de compras a partir de las tablas **company** y **transaction**, unidas mediante un JOIN por el id de la compañía. Agrupamos los resultados por los datos que deseamos mostrar y los ordenamos de forma descendente por el promedio.



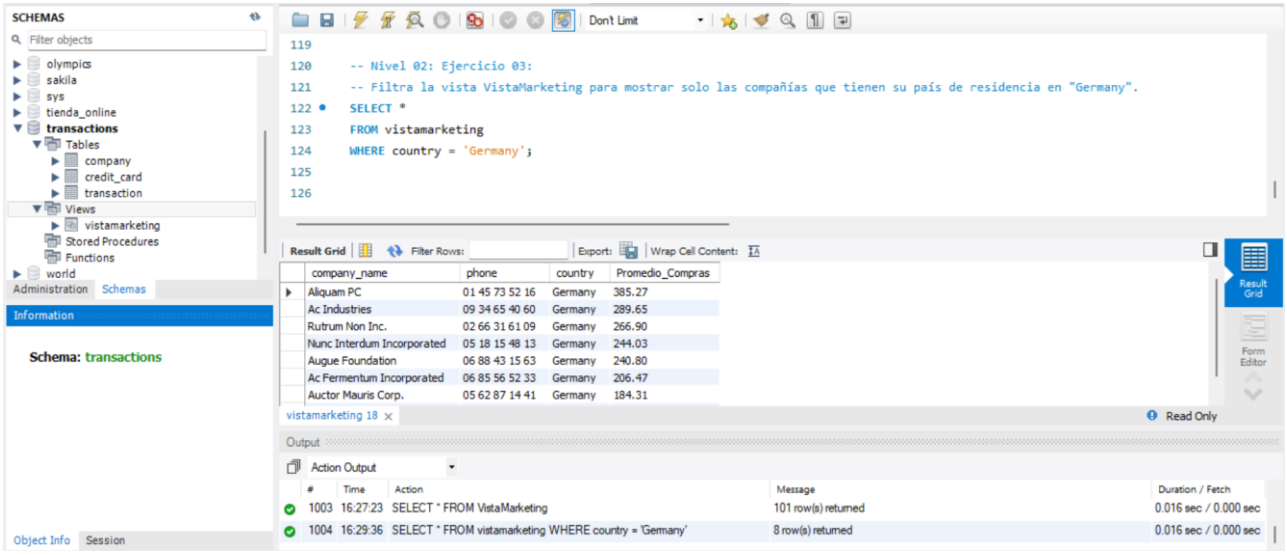
Como resultados, tenemos la vista creada, y a la lateral izquierda aparece la creación de la vista.

** En la entrega anterior, faltaba el ON en el JOIN, y los promedios estaban igual. Ahora se ha solucionado.*

EJERCICIO 3

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

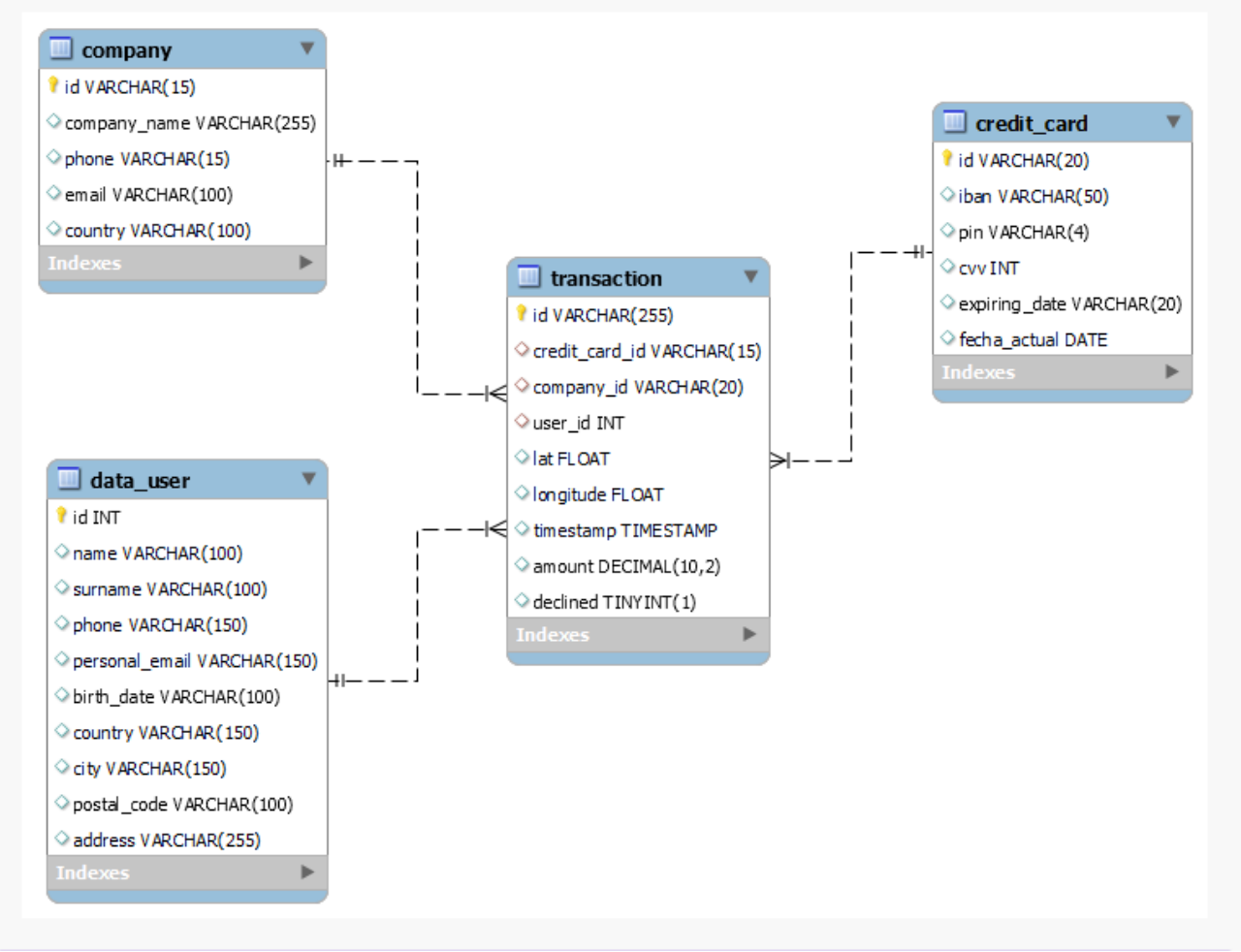
En este caso, la vista funciona como una nueva tabla y hacemos una consulta directamente en ella, utilizando el filtro **WHERE** para encontrar el país deseado.



NIVEL 3

EJERCICIO 01

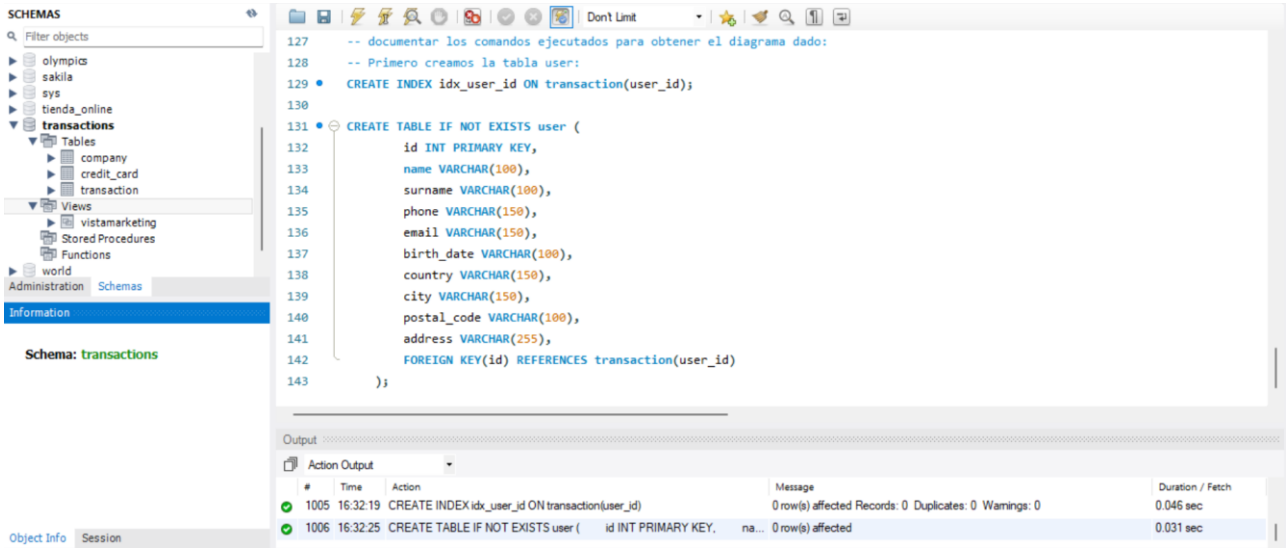
La próxima semana tendrás una reunión con los gerentes de marketing. Un compañero de tu equipo realizó modificaciones en la base de datos, pero no recuerda cómo las realizó. Te pide que lo ayudes a documentar los comandos ejecutados para obtener el siguiente diagrama:



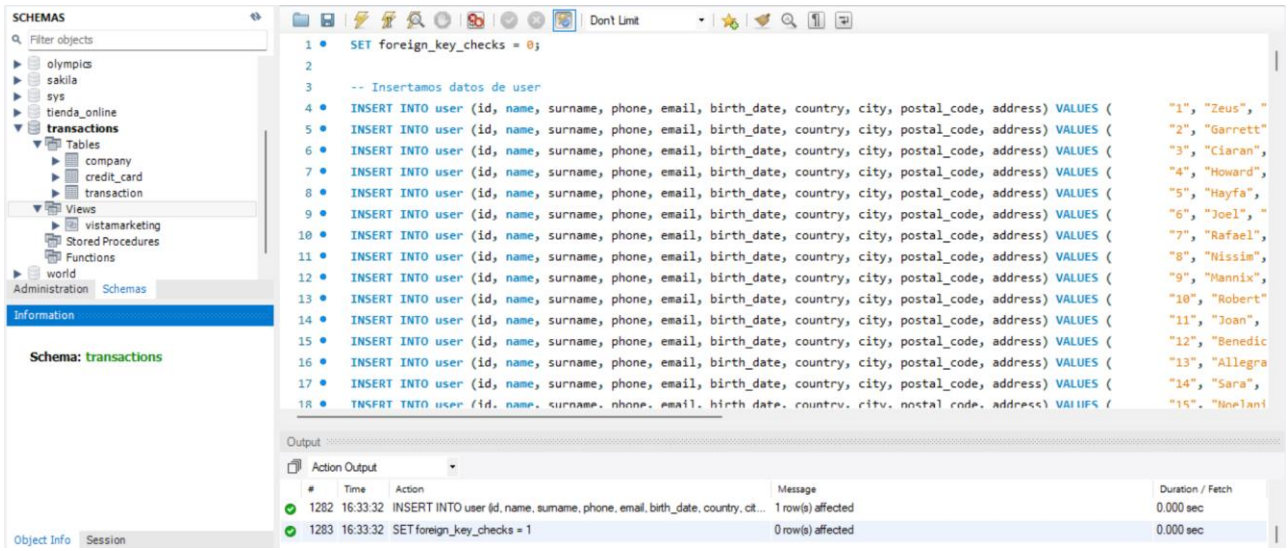
Recordatorio

En esta actividad, es necesario que describas el "paso a paso" de las tareas realizadas. Es importante realizar descripciones sencillas, simples y fáciles de comprender. Para llevar a cabo esta actividad, deberás trabajar con los archivos denominados *"estructura_dades_user"* y *"dades_introduir_user"*.

Para ejecutar esta tarea, inicialmente, creamos la tabla user, que ya tenemos la estructura ordenada:

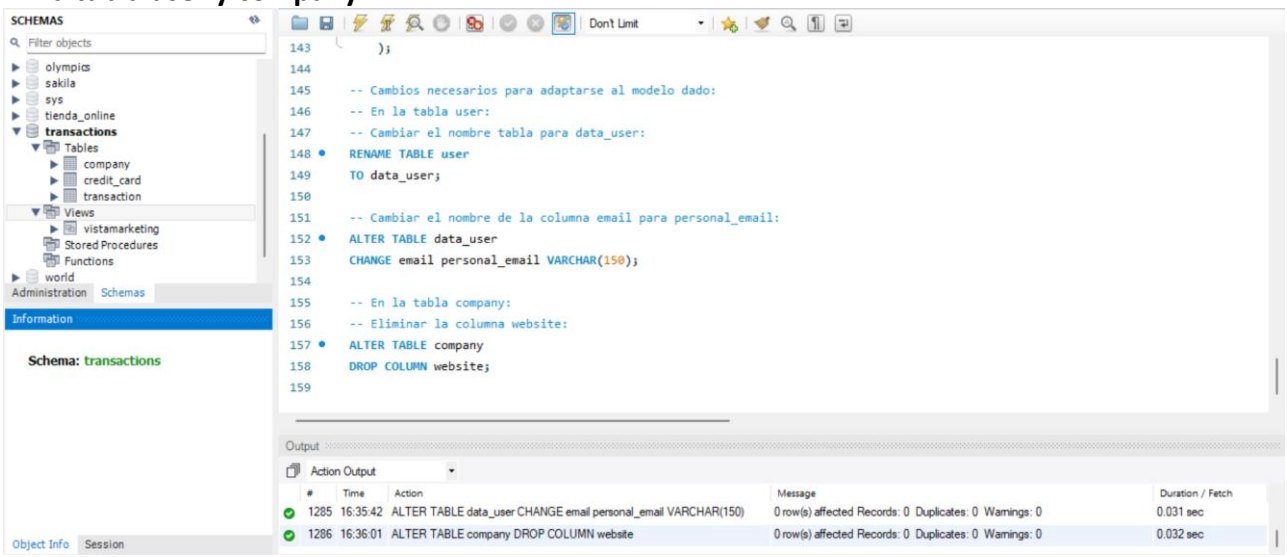


Luego, inserimos los datos:

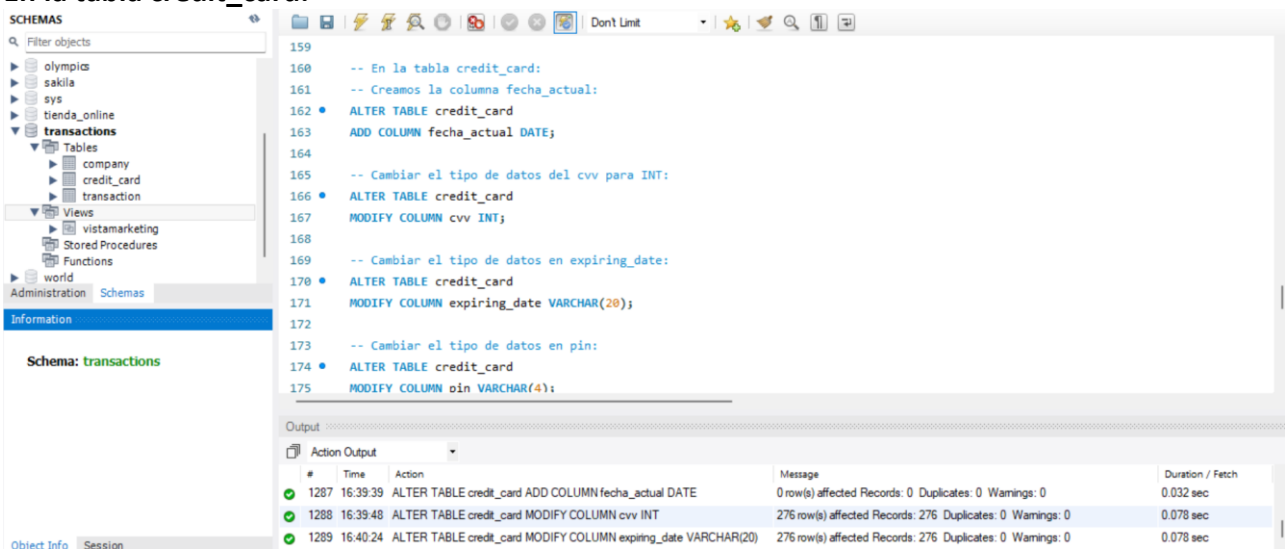


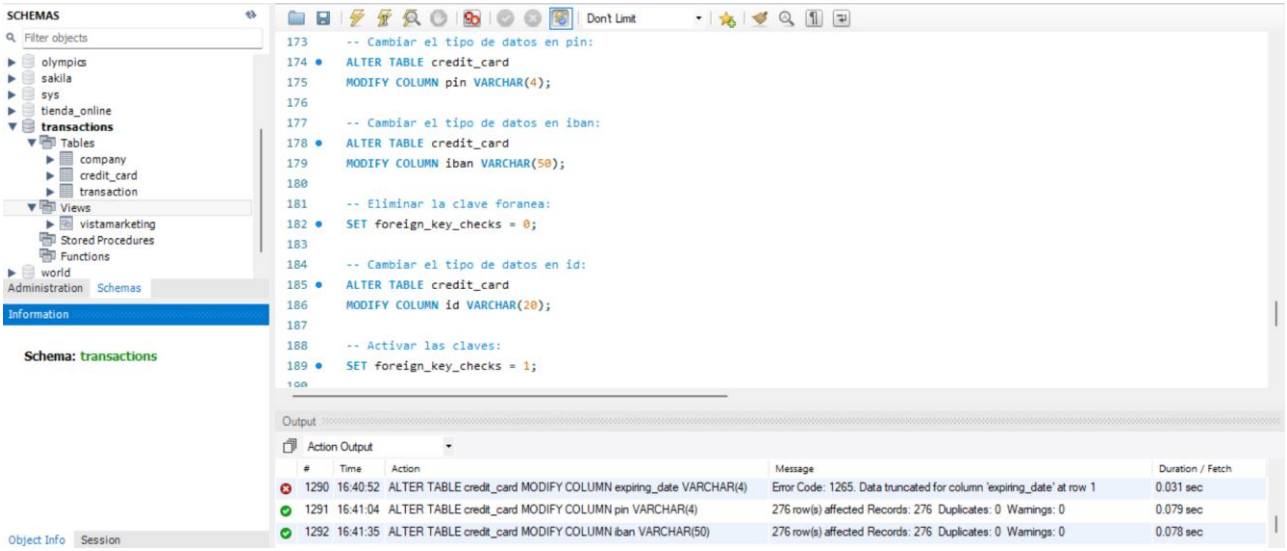
Para adaptar las tablas con el modelo referenciado inicialmente es necesario ajustar algunas columnas y tablas:

En la tabla user y company:

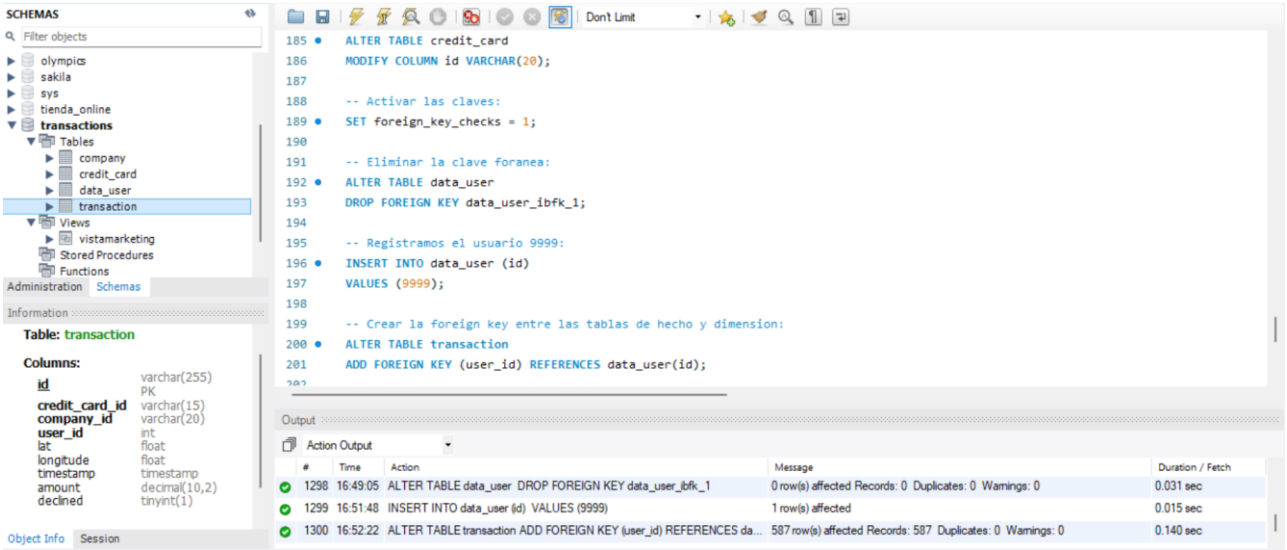


En la tabla credit_card:





Falta ajustar la **foreing key** de **user_id** en **transaction** e id en **data_user.** Para corregir la relación de tipo 1 a N, será necesario eliminar la relación actual y agregar el registro previamente creado en la tabla transaction con user_id = 9999. Si los datos no coinciden y existen más registros en transaction que no tienen correspondencia en la tabla data_user, se generará una relación inversa, como la que tenemos ahora, donde la cardinalidad es de 1 a N de transaction hacia data_user.



Con estos cambios, comparamos los diagramas:

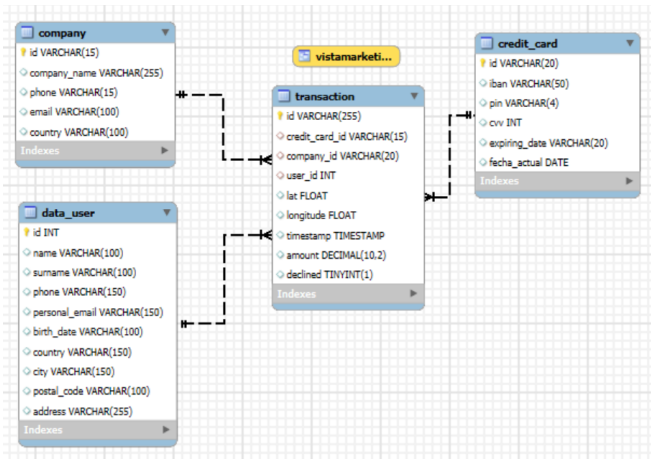


Diagrama creado

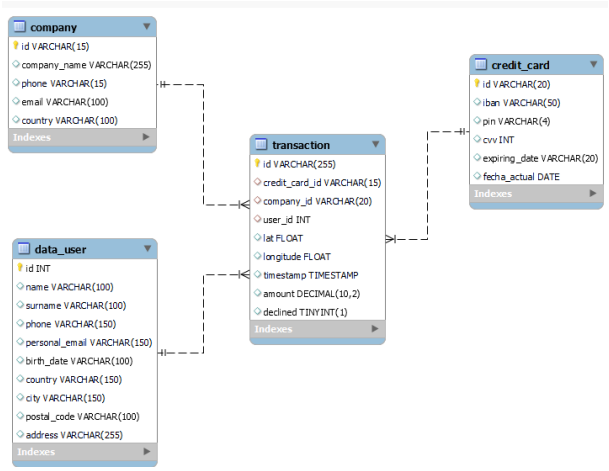
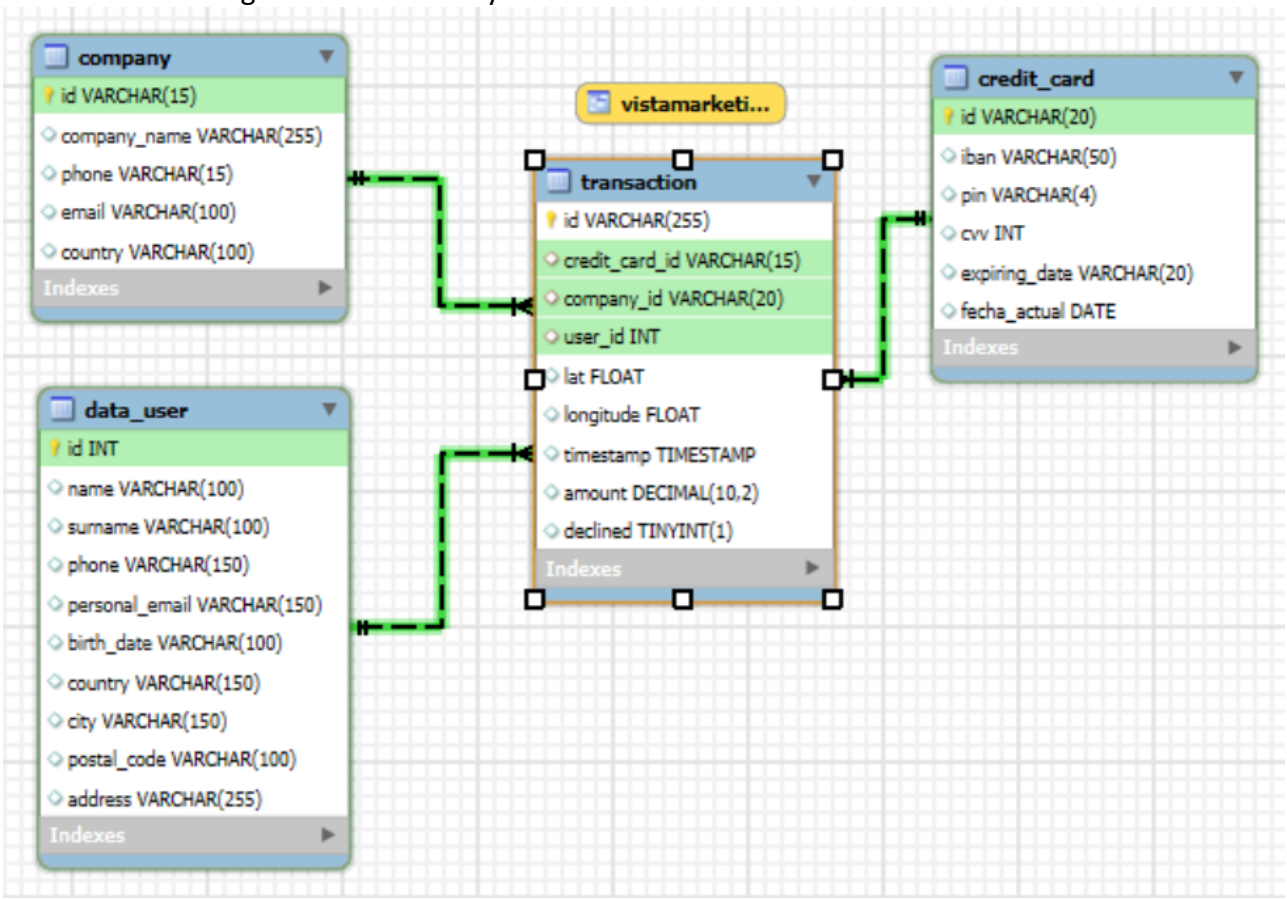


Diagrama dado para el ejercicio

Así tenemos el diagrama actualizado y con las cardinalidades correctas:



Así, al final tenemos un diagrama, donde la tabla de hechos es transaction y las de dimensiones company, credit_card y data_user, que fue creada nueva.

EJERCICIO 02

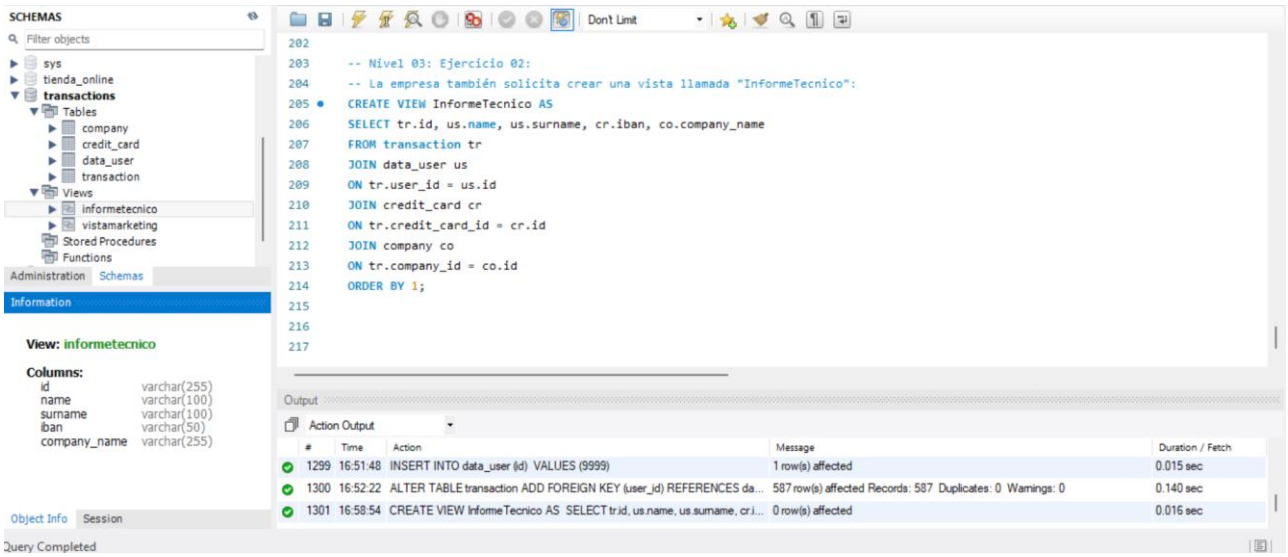
La empresa también solicita crear una vista llamada *"InformeTecnico"* que contenga la siguiente información:

- 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.

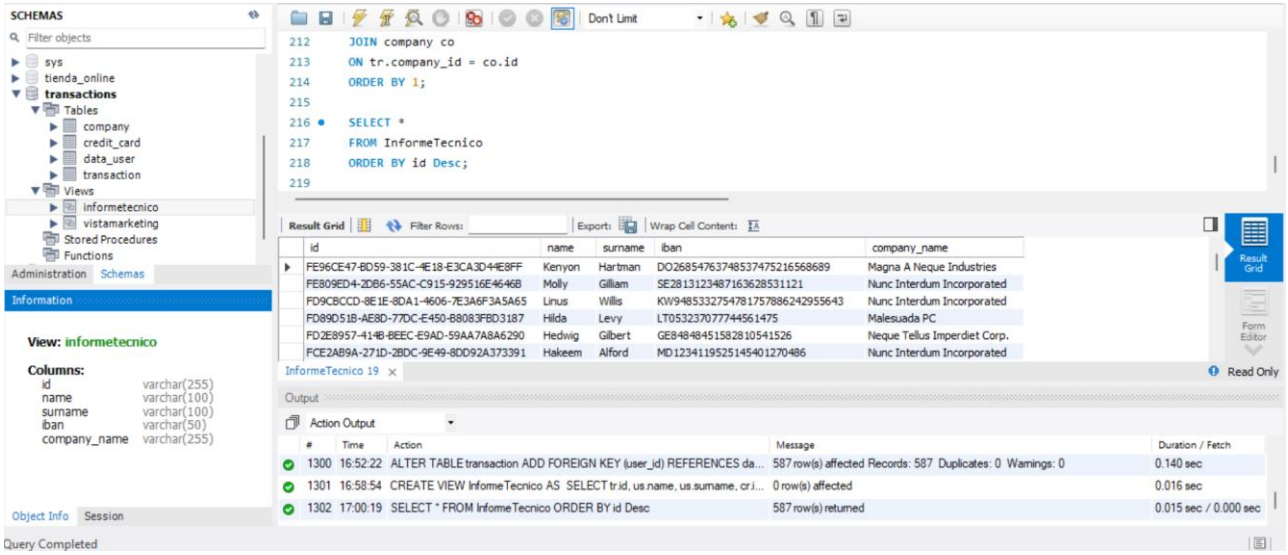
Asegúrate de incluir información relevante de ambas tablas y utiliza alias para renombrar columnas según sea necesario.

Muestra los resultados de la vista y ordena los resultados de manera descendente en función de la variable *ID de transaction*.

Generamos la vista utilizando **CREATE VIEW** e incluimos todas las variables solicitadas, para conectar las tablas **data_user**, **credit_card** y **company** con **transaction**, realizamos tres uniones (JOIN).



Luego, visualizamos el contenido de la vista ejecutando **SELECT * FROM InformeTecnico,** organizamos los resultados de forma descendente (usando DESC) según el id de la tabla **transaction**, para mostrar los valores de mayor a menor.


SCHEMAS
Filter objects
sys
tienda_online
transactions
Tables
company
credit_card
data_user
transaction
Views
informetecnico
vistamarketing
Stored Procedures
Functions
Administration
Schemas
Information
View: informetecnico
Columns:
id varchar(255)
name varchar(100)
surname varchar(100)
iban varchar(50)
company_name varchar(255)
Object Info
Session
Don't Limit
JOIN company co
ON tr.company_id = co.id
ORDER BY 1;
SELECT *
FROM InformeTecnico
ORDER BY id Desc;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
id
name
surname
iban
company_name
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF
Kenyon
Hartman
DO26854763748537475216568689
Magna A Neque Industries
FE809ED4-2DB6-55AC-C915-929516E4646B
Molly
Gilliam
SE2813123487163628531121
Nunc Interdum Incorporated
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65
Linus
Willis
KW9485332754781757886242955643
Nunc Interdum Incorporated
FD89D51B-AE8D-77DC-E450-B8083FBD3187
Hilda
Levy
LT053237077744561475
Malesuada PC
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290
Hedwig
Gilbert
GE84848451582810541526
Neque Tellus Imperdiet Corp.
FCE2AB9A-271D-2BDC-9E49-8DD92A373391
Hakeem
Alford
MD1234119525145401270486
Nunc Interdum Incorporated
InformeTecnico 19
Result Grid
Form Editor
Read Only
Output
Action Output
Time Action Message Duration / Fetch
1300 16:52:22 ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES da... 587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0 0.140 sec
1301 16:58:54 CREATE VIEW InformeTecnico AS SELECT tr.id, us.name, us.surname, cr.i... 0 row(s) affected 0.016 sec
1302 17:00:19 SELECT * FROM InformeTecnico ORDER BY id Desc 587 row(s) returned 0.015 sec / 0.000 sec
Query Completed