

# DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

## SPRINT 03

Manipulación de tablas  
30 de abril de 2025



Letiane Benincá  
benincalf@gmail.com

IT ACADEMY



## 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(15):** clave primaria, alfanuméricos de hasta 15 caracteres. ['CcU-2938']
- **iban VARCHAR(50):** International Bank Account Number, alfanuméricos de hasta 50 caracteres. ['TR301950312213576817638661']
- **pan VARCHAR(20):** Primary Account Number, almacena el número de la tarjeta, alfanuméricos de hasta 20 caracteres. ['5424465566813633']
- **pin VARCHAR(4):** Código secreto de la tarjeta, alfanuméricos de hasta 4 caracteres. ['3257']
- **cvv VARCHAR(3):** Card Verification Value, alfanuméricos de hasta 3 caracteres. ['984']
- **expiring\_date VARCHAR(10):** 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.

**Consulta:**

```
CREATE TABLE IF NOT EXISTS credit_card (  
    id VARCHAR(15) PRIMARY KEY,  
    iban VARCHAR(50),  
    pan VARCHAR(20),  
    pin VARCHAR(4),  
    cvv VARCHAR(3),  
    expiring_date DATE,  
    FOREIGN KEY (id)  
    REFERENCES transaction (credit_card_id));
```

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\_credito".

```

30042025_LB_Data Analytics_S... x datos_introducir_credit estructura_datos_user
1  -- Nivel 01: Ejercicio 01:
2  -- Diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito.
3  CREATE TABLE IF NOT EXISTS credit_card (
4      id VARCHAR(15) PRIMARY KEY,
5      iban VARCHAR(50),
6      pan VARCHAR(20),
7      pin VARCHAR(4),
8      cvv VARCHAR(3),
9      expiring_date DATE,
10     FOREIGN KEY (id)
11     REFERENCES transaction (credit_card_id)
12 );

```

La tabla con los datos inseridos:

```

30042025_LB_Data Analytics_S... x
7      pin VARCHAR(4),
8      cvv VARCHAR(3),
9      expiring_date VARCHAR(10),
10     FOREIGN KEY (id)
11     REFERENCES transaction (credit_card_id)
12 );
13
14 SELECT *
15 FROM credit_card;

```

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	10/30/22
CcU-2945	DO26854763748537475216568689	5142423821948828	9080	887	08/24/23
CcU-2952	BG45IVQL52710525608255	4556 453 55 5287	4598	438	06/29/21
CcU-2959	CR7242477244335841535	372461377349375	3583	667	02/24/23
CcU-2966	BG72LKTQ15627628377363	448566 886747 7265	4900	130	10/29/24
CcU-2973	PT87806228135092429456346	544 58654 54343 384	8760	887	01/30/25
CcU-2980	DE39241881883086277136	402400 7145845969	5075	596	07/24/22
CcU-2987	GE89681434837748781813	3763 747687 76666	2298	797	10/31/23
CcU-2994	BH62714428368066765294	344283273252593	7545	595	02/28/22

credit\_card 8 x Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
989	13:27:20	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( ...	1 row(s) affected	0.000 sec

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:

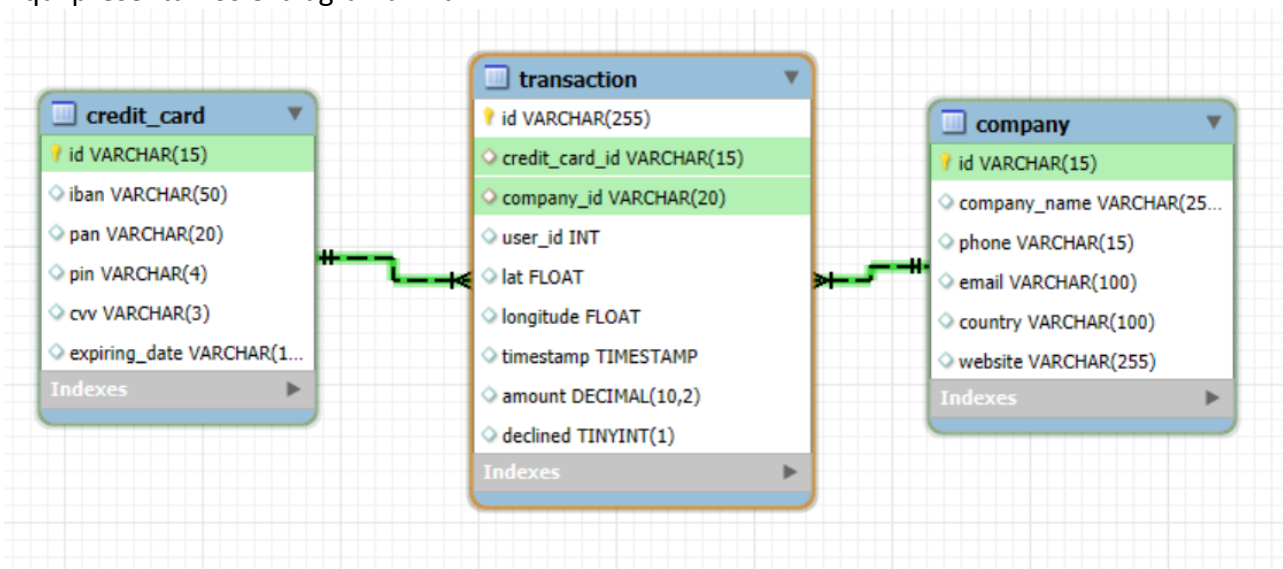
**Consulta:** ALTER TABLE transaction  
ADD CONSTRAINT fk\_transaction\_credit\_card  
FOREIGN KEY (credit\_card\_id)  
REFERENCES credit\_card(id);

```

30042025_LB_Data_Analytics_S...
2  -- Diseñar y crear una tabla llamada "credit_card" que almacene detalles cruciales sobre las tarjetas de crédito.
3  CREATE TABLE IF NOT EXISTS credit_card (
4      id VARCHAR(15) PRIMARY KEY,
5      iban VARCHAR(50),
6      pan VARCHAR(20),
7      pin VARCHAR(4),
8      cvv VARCHAR(3),
9      expiring_date VARCHAR(10),
10     FOREIGN KEY (id)
11     REFERENCES transaction (credit_card_id)
12 );
13
14 ALTER TABLE transaction
15 ADD CONSTRAINT fk_transaction_credit_card
16 FOREIGN KEY (credit_card_id)
17 REFERENCES credit_card(id);

```

Aquí 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:

The screenshot shows a database management tool interface. The top panel displays a SQL query:

```

21
22 -- Nivel 01: Ejercicio 02:
23 -- El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938.
24 -- La información que debe mostrarse para este registro es: R323456312213576817699999.
25 • SELECT *
26 FROM credit_card
27 WHERE id = 'CcU-2938';
28

```

The middle panel shows the 'Result Grid' with the following data:

	id	iban	pin	cvv	expiring_date	fecha_actual
▶	CcU-2938	R323456312213576817699999	3257	984	10/30/22	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL

The bottom panel shows the 'Output' section with the following action output:

#	Time	Action	Message	Duration / Fetch
6	10:34:54	SELECT * FROM InformeTecnico ORDER BY id Desc	587 row(s) returned	0.094 sec / 0.000 sec
7	10:43:19	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

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

**Consulta:** UPDATE credit\_card  
 SET iban = 'R323456312213576817699999'  
 WHERE id = 'CcU-2938';

Tenemos como resultado:

30042025\_IB\_Data Analytics\_S...

```

29 • UPDATE credit_card
30 SET iban = 'R32345631221357681769999'
31 WHERE id = 'CcU-2938';
32
33 • SELECT *
34 FROM credit_card
35 WHERE id = 'CcU-2938';
36

```

Result Grid

id	iban	pin	cvv	expiring_date	fecha_actual
CcU-2938	R32345631221357681769999	3257	984	10/30/22	NULL
NULL	NULL	NULL	NULL	NULL	NULL

credit\_card 4

Output

Action Output

#	Time	Action	Message	Duration / Fetch
7	10:43:19	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec
8	10:44:49	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec

## EJERCICIO 3

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

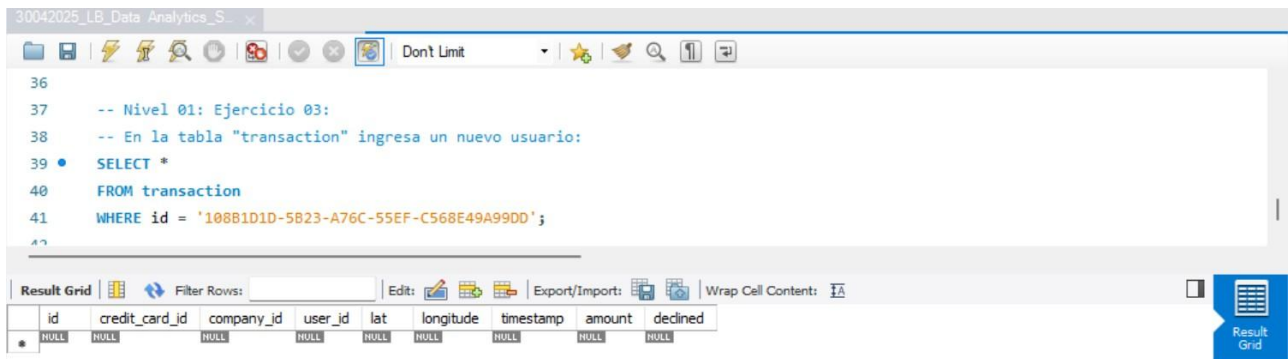
Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lat	829.999
longitude	-117.999
amount	111.11
declined	0

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

```

SELECT *
FROM transaction
WHERE id = '108B1D1D-5B23-A76C-55EF-C568E49A99DD';

```



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

```

INSERT INTO company (id)
VALUES ('b-9999');

```

```

INSERT INTO credit_card (id)
VALUES ('CcU-9999');

```

```

INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount,
declined)
VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', 9999,
829.999, -117.999, 111.11, 0);

```

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.



The screenshot shows a SQL IDE window titled "30042025\_LB\_Data Analytics\_S...". The query editor contains the following SQL code:

```

43 • INSERT INTO company (id)
44   VALUES ('b-9999');
45
46 • INSERT INTO credit_card (id)
47   VALUES ('CcU-9999');
48
49 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
50   VALUES ('10881D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', 9999, 829.999, -117.999, 111.11, 0);
51
52 • SELECT *
53   FROM transaction
54  WHERE id = '10881D1D-5B23-A76C-55EF-C568E49A99DD';

```

The "Result Grid" shows the following data:

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
▶	10881D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	NULL	111.11	0
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The "Output" window shows the "Action Output" for the query execution:

#	Time	Action	Message	Duration / Fetch
8	10:44:49	SELECT * FROM credit_card WHERE id = 'CcU-2938'	1 row(s) returned	0.000 sec / 0.000 sec
9	10:47:03	SELECT * FROM transaction WHERE id = '10881D1D-5B23-A76C-55EF-C5...	1 row(s) returned	0.000 sec / 0.000 sec

## EJERCICIO 4

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

Para esto, utilizamos el `ALTER TABLE credit_card DROP COLUMN pan` eliminamos la columna.

```

ALTER TABLE credit_card
DROP COLUMN pan;

```



30042025\_LB\_Data Analytics\_S... dades\_introduir

55  
56 -- Nivel 01: Ejercicio 04:  
57 -- Desde Recursos Humanos solicitan eliminar la columna "pan" de la tabla credit\_card.  
58 • ALTER TABLE credit\_card  
59 DROP COLUMN pan;  
60  
61 • SELECT \*  
62 FROM credit\_card;  
63

Result Grid

	id	iban	pin	cvv	expiring_date
▶	CcU-2938	R32345631221357681769999	3257	984	10/30/22
	CcU-2945	DO26854763748537475216568689	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

credit\_card 9 x Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 22	14:33:54	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
✓ 23	14:34:12	SELECT * FROM credit_card	276 row(s) returned	0.016 sec / 0.000 sec

Así queda el resultado, la tabla credit\_card ya no tiene la columna pan.

## 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:

The screenshot shows a data analytics interface with a SQL editor and a result grid. The SQL editor contains the following code:

```

63
64 -- Nivel 02: Ejercicio 01:
65 -- Elimina de la tabla transaction el registro con ID 02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de datos.
66 • SELECT *
67 FROM transaction
68 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
69
70
71

```

The result grid displays the following data:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	92	81.9185	-12.5276	2021-08-28 23:42:24	466.92	0

Below the result grid, the 'transaction 10' tab is active, showing the 'Output' section with the following action output:

#	Time	Action	Message	Duration / Fetch
23	14:34:12	SELECT * FROM credit_card	276 row(s) returned	0.016 sec / 0.000 sec
24	14:39:07	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88...	1 row(s) returned	0.000 sec / 0.000 sec

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.

30042025\_LB\_Data Analytics\_S...

```

67 FROM transaction
68 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
69
70 • DELETE FROM transaction
71 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
72
73 • SELECT *
74 FROM transaction
75 WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';

```

Result Grid

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 11 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 25	14:41:39	DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D...	1 row(s) affected	0.000 sec
✓ 26	14:42:03	SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88...	0 row(s) returned	0.000 sec / 0.000 sec

## 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.

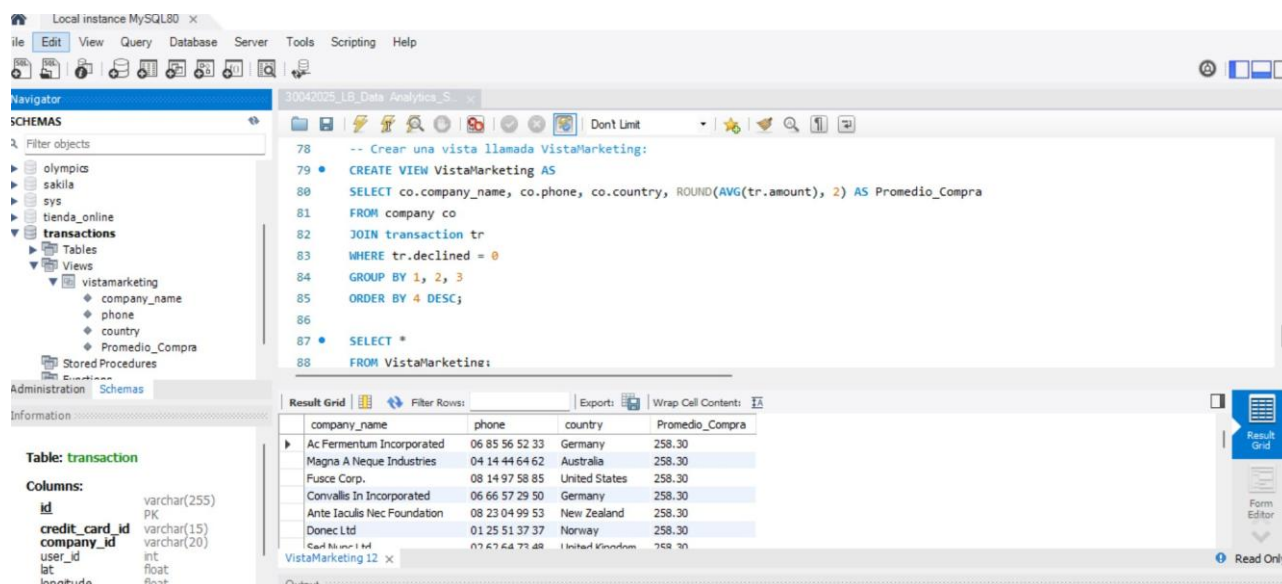
Usamos el comando **CREATE VIEW AS** para crear la vista. Seleccionamos el nombre de la compañía, el teléfono y el país, además de calcular el promedio de compras a partir de las tablas **company** y **transaction**, unidas mediante un JOIN por el id de la compañía. Agrupamos los resultados por los datos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

```

CREATE VIEW VistaMarketing AS
SELECT co.company_name, co.phone, co.country, ROUND(AVG(tr.amount), 2) AS
Promedio_Compra
FROM company co
JOIN transaction tr

```

```
WHERE tr.declined = 0
GROUP BY 1, 2, 3
ORDER BY 4 DESC;
```



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

## 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.

```
SELECT *
FROM vistamarketing
WHERE country = 'Germany';
```

30042025\_LB\_Data\_Analytics\_S\_ x

Don't Limit

```

89
90 -- Nivel 02: Ejercicio 03:
91 -- Filtra la vista VistaMarketing para mostrar solo las compañías que tienen su país de residencia en "Germany".
92 • SELECT *
93 FROM vistamarketing
94 WHERE country = 'Germany';
95
96
97
98

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:  $\frac{1}{A}$

company_name	phone	country	Promedio_Compra
Rutrum Non Inc.	02 66 31 61 09	Germany	258.30
Aliquam PC	01 45 73 52 16	Germany	258.30
Auctor Mauris Corp.	05 62 87 14 41	Germany	258.30
Ac Industries	09 34 65 40 60	Germany	258.30
Augue Foundation	06 88 43 15 63	Germany	258.30
Nunc Interdum Incorporated	05 18 15 48 13	Germany	258.30

vistamarketing 14 x

Read Only

Output

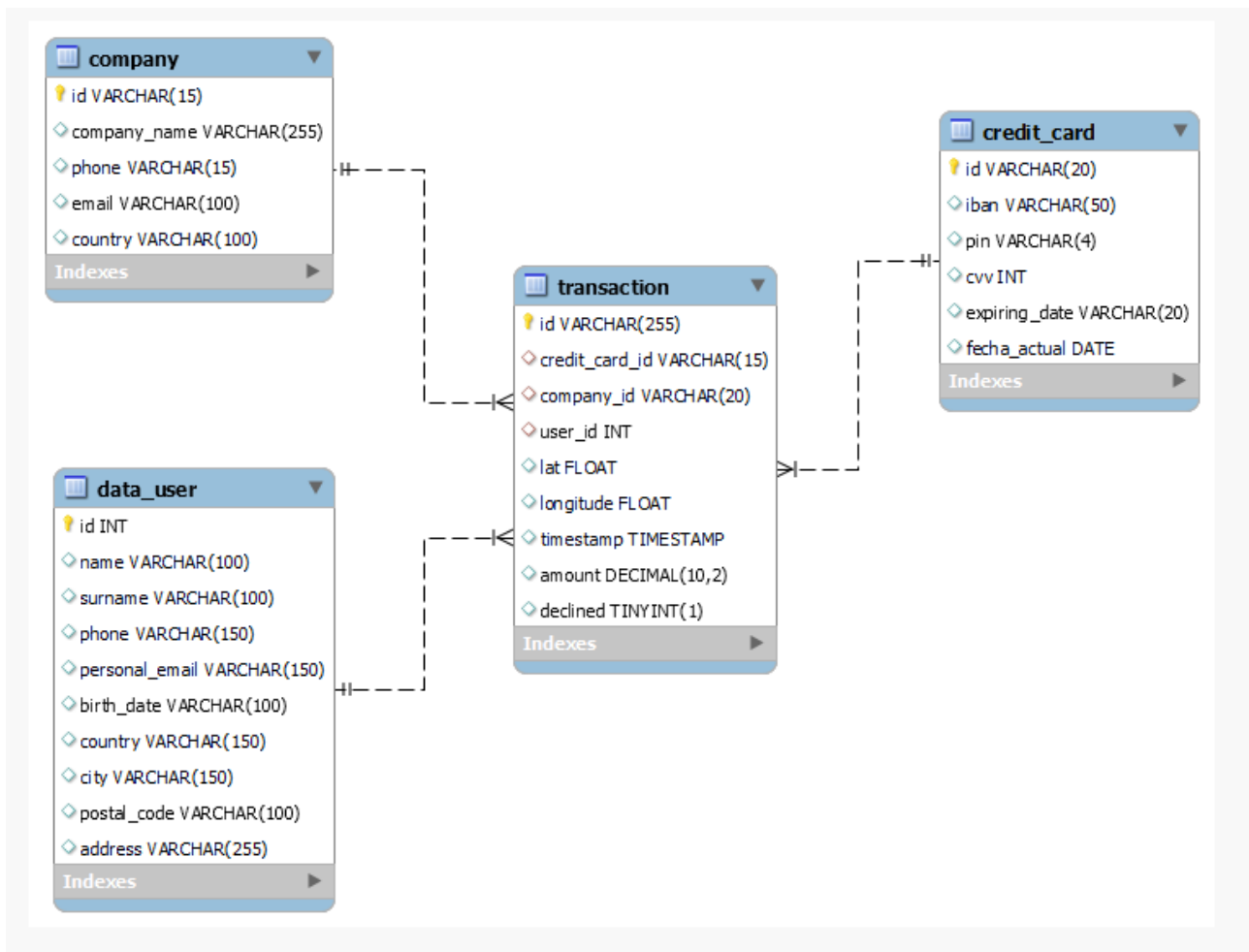
Action Output

#	Time	Action	Message	Duration / Fetch
29	14:53:34	SELECT company_name, country FROM vistamarketing WHERE country = '...	8 row(s) returned	0.016 sec / 0.000 sec

## 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:

```

30042025_LB_Data Analytics_S... | datos_introducir_user (1) | estructura_datos_user
-- documentar los comandos ejecutados para obtener el diagrama dado:
-- Primero creamos la tabla user:
99 • CREATE INDEX idx_user_id ON transaction(user_id);
100
101 • CREATE TABLE IF NOT EXISTS user (
102     id INT PRIMARY KEY,
103     name VARCHAR(100),
104     surname VARCHAR(100),
105     phone VARCHAR(150),
106     email VARCHAR(150),
107     birth_date VARCHAR(100),
108     country VARCHAR(150),
109     city VARCHAR(150),
110     postal_code VARCHAR(100),
111     address VARCHAR(255),
112     FOREIGN KEY(id) REFERENCES transaction(user_id)
113 );
114
Output
Action Output
# Time Action Message Duration / Fetch
31 15:01:30 CREATE INDEX idx_user_id ON transaction(user_id) 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 0.078 sec

```

Luego, inserimos los datos:

```

30042025_LB_Data Analytics_S... | datos_introducir_user (1) |
1 • SET foreign_key_checks = 0;
2
3 -- Insertamos datos de user
4 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "1", "Zeus", "
5 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "2", "Garrett"
6 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "3", "Ciaran",
7 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "4", "Howard",
8 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "5", "Hayfa",
9 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "6", "Joel", "
10 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "7", "Rafael",
11 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "8", "Nissim",
12 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "9", "Mannix",
13 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "10", "Robert"
14 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "11", "Joan",
15 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "12", "Benedic
16 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "13", "Allegra
17 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "14", "Sara",
18 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES ( "15", "Noelani

Output
Action Output
# Time Action Message Duration / Fetch
308 15:03:17 INSERT INTO user (id, name, surname, phone, email, birth_date, country, cit... 1 row(s) affected 0.000 sec

```

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

En la tabla user:



*Cambiar el nombre tabla para data\_user:*

```
RENAME TABLE user TO data_user;
```

*Cambiar el nombre de la columna email para personal\_email:*

```
ALTER TABLE data_user  
CHANGE email personal_email VARCHAR(150);
```

**En la tabla company:**

*Eliminar la columna website:*

```
ALTER TABLE company  
DROP COLUMN website;
```

**En la tabla credit\_card:**

*Creamos la columna fecha\_actual:*

```
ALTER TABLE credit_card  
ADD COLUMN fecha_actual DATE;
```

**Cambiar el tipo de datos del cvv para INT:**

```
ALTER TABLE credit_card  
MODIFY COLUMN cvv INT;
```

**Cambiar el tipo de datos en expiring\_date:**

```
ALTER TABLE credit_card  
MODIFY COLUMN expiring_date VARCHAR(20);
```

Con estos cambios, comparamos los diagramas:

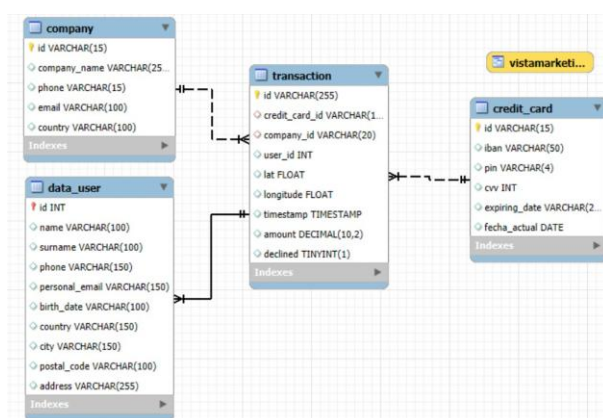


Diagrama creado

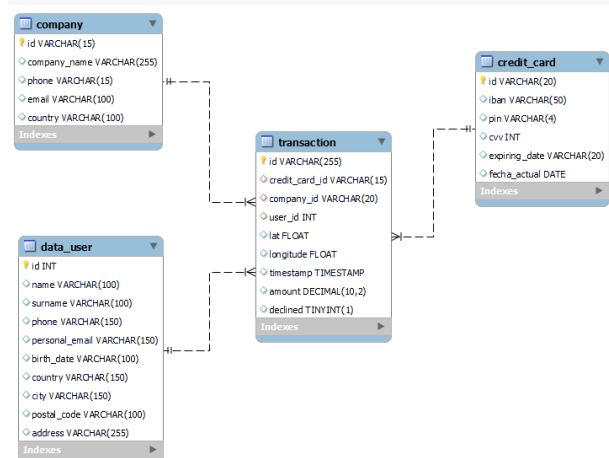


Diagrama dado para el ejercicio

En este sentido, falta ajustar la **foreign 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.

Inicialmente eliminamos la clave foránea.

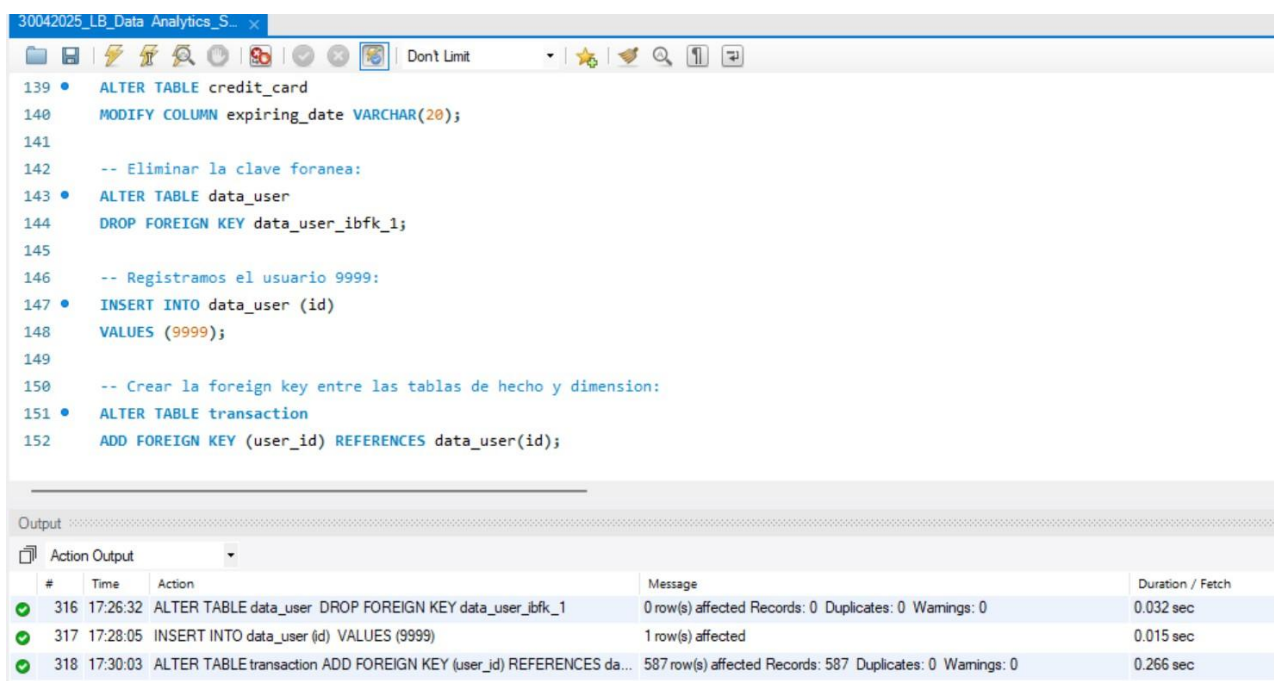
```
ALTER TABLE data_user
DROP FOREIGN KEY data_user_ibfk_1;
```

Registramos el usuario 9999:

```
INSERT INTO data_user (id)
VALUES (9999);
```

Crear la foreign key entre las tablas de hecho y dimensión:

```
ALTER TABLE transaction
ADD FOREIGN KEY (user_id) REFERENCES data_user(id);
```

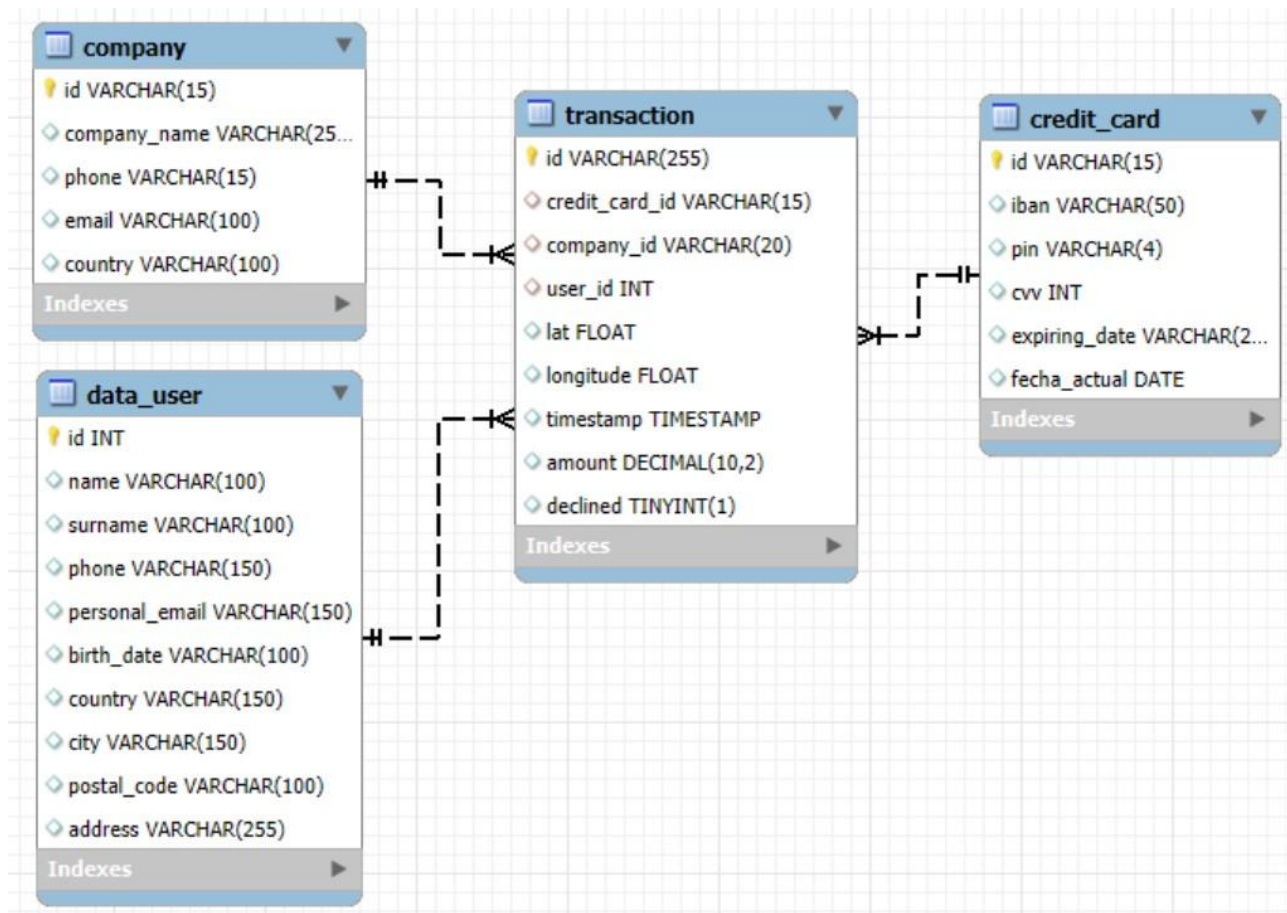


```
30042025_LB_Data Analytics S... x
ALTER TABLE credit_card
MODIFY COLUMN expiring_date VARCHAR(20);
-- Eliminar la clave foranea:
ALTER TABLE data_user
DROP FOREIGN KEY data_user_ibfk_1;
-- Registramos el usuario 9999:
INSERT INTO data_user (id)
VALUES (9999);
-- Crear la foreign key entre las tablas de hecho y dimension:
ALTER TABLE transaction
ADD FOREIGN KEY (user_id) REFERENCES data_user(id);
```

Output

#	Time	Action	Message	Duration / Fetch
316	17:26:32	ALTER TABLE data_user DROP FOREIGN KEY data_user_ibfk_1	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.032 sec
317	17:28:05	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected	0.015 sec
318	17:30:03	ALTER TABLE transaction ADD FOREIGN KEY (user_id) REFERENCES data_user(id)	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	0.266 sec

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).

```
CREATE VIEW InformeTecnico AS
SELECT tr.id, us.name, us.surname, cr.iban, co.company_name
FROM transaction tr
JOIN data_user us
ON tr.user_id = us.id
JOIN credit_card cr
ON tr.credit_card_id = cr.id
JOIN company co
ON tr.company_id = co.id
ORDER BY 1;
```

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane shows the database structure, including tables and views. The 'transaction' table is highlighted. The main pane shows the SQL script for creating the 'InformeTecnico' view, which joins the 'transaction', 'data\_user', 'credit\_card', and 'company' tables. Below the script, the 'Result Grid' shows the output of the view, displaying columns: id, name, surname, iban, and company\_name. The results are sorted by id in descending order. The bottom pane shows the 'Output' window with a message indicating that the view was successfully created and that 587 rows were affected.

id	name	surname	iban	company_name
0466A42E-47CF-8D24-FD01-C06689713128	William	Benjamin	MD1234119525145401270486	Nunc Interdum Incorporated
063FBA79-99EC-66FB-29F7-25726D1764A5	Kenyon	Hartman	GE89681434837748781813	Amet Nulla Donec Corporation
0668296C-CD89-A883-76BC-2E4C4F8C8AE	Chloe	Keith	BA542358041365401657	Non Institute
06CD9AA5-9B42-D684-DDDD-A5E394FEB999	Lynn	Riddle	CR7242477244335841535	Ut Semper Foundation
07A46D48-31A3-7E87-65B9-0DA902AD109F	Hedwig	Gilbert	F19398462343991818	Lacus Quisque Associates
09DE92CE-6F27-2B87-13B5-938582B388E2	Kenyon	Hartman	NO8923814763512	Elit Etiam Laoreet Associates

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.

```
SELECT *
FROM InformeTecnico
ORDER BY id DESC;
```

30042025\_LB\_Data\_Analytics\_S...

```

158 FROM transaction tr
159 JOIN data_user us
160 ON tr.user_id = us.id
161 JOIN credit_card cr
162 ON tr.credit_card_id = cr.id
163 JOIN company co
164 ON tr.company_id = co.id
165 ORDER BY 1;
166
167 • SELECT *
168 FROM InformeTecnico
169 ORDER BY id Desc;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

id	name	surname	iban	company_name
0466A42E-47CF-8D24-FD01-C0B689713128	William	Benjamin	MD1234119525145401270486	Nunc Interdum Incorporated
063FBA79-99EC-66FB-29F7-25726D1764A5	Kenyon	Hartman	GE89681434837748781813	Amet Nulla Donec Corporation
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	Chloe	Keith	BA542358041365401657	Non Institute
06CD9AA5-9B42-D684-DDDD-A5E394FEB9A99	Lynn	Riddle	CR7242477244335841535	Ut Semper Foundation
07A46D48-31A3-7E87-65B9-0DA902AD109F	Hedwig	Gilbert	FI9398462343991818	Lacus Quisque Associates

InformeTecnico 6 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
9	10:47:03	SELECT * FROM transaction WHERE id = '108B1D1D-5B23-A76C-55EF-C5...	1 row(s) returned	0.000 sec / 0.000 sec
10	10:52:20	SELECT * FROM InformeTecnico	587 row(s) returned	0.015 sec / 0.000 sec