

# IT ACADEMY

## SPRINT 3: **Manipulació de taules**

Sergi Alcolea de la Gala

# Nivell 1

## Exercici 1

La teva tasca és dissenyar i crear una taula anomenada "credit\_card" que emmagatzemi detalls crucials sobre les targetes de crèdit. La nova taula ha de ser capaç d'identificar de manera única cada targeta i establir una relació adequada amb les altres dues taules ("transaction" i "company"). Després de crear la taula serà necessari que ingressis la informació del document denominat "dades\_introduir\_credit". Recorda mostrar el diagrama i realitzar una breu descripció d'aquest.

```
CREATE TABLE IF NOT EXISTS credit_card (    -- IF NOT EXISTS serve para
asegurarse de que no estás creando dos veces la misma tabla.
    id VARCHAR(34) PRIMARY KEY,
    iban VARCHAR(34) NOT NULL,
    pan VARCHAR(34) NOT NULL,
    pin VARCHAR(4) NOT NULL,
    cvv VARCHAR(3) NOT NULL,
    expiring_date VARCHAR(10) NOT NULL
);
```

The screenshot shows a database IDE interface. The SQL editor contains the following code:

```
26 • USE transactions;
27 • CREATE TABLE IF NOT EXISTS credit_card (    -- IF NOT EXISTS serve para asegurarse de que no estás creando do
28     id VARCHAR(34) PRIMARY KEY,
29     iban VARCHAR(34) NOT NULL,
30     pan VARCHAR(34) NOT NULL,
31     pin VARCHAR(4) NOT NULL,
32     cvv VARCHAR(3) NOT NULL,
33     expiring_date VARCHAR(10) NOT NULL
34 );
35 • SELECT * FROM credit_card;
36
```

Below the editor, the 'Result Grid' shows a single row with all columns (id, iban, pan, pin, cvv, expiring\_date) containing NULL values.

The 'Output' pane at the bottom shows the execution log:

#	Time	Action	Message	Duration / Fetch
✓ 100142	13:17:38	CREATE TABLE IF NOT EXISTS credit_card (- ...	0 row(s) affected	0.015 sec
✓ 100143	13:17:46	SELECT * FROM credit_card	0 row(s) returned	0.000 sec / 0.000 sec

→ Para determinar qué campos debía introducir en la tabla de credit\_card, me basé en los datos presentes en el documento "datos\_introducir\_sprint3\_credit", que eran: id, iban, pan, pin, cvv y la fecha de caducidad. Usé el comando "IF NOT EXISTS" en crate para asegurarme de que esta tabla no se duplica en la base de datos.

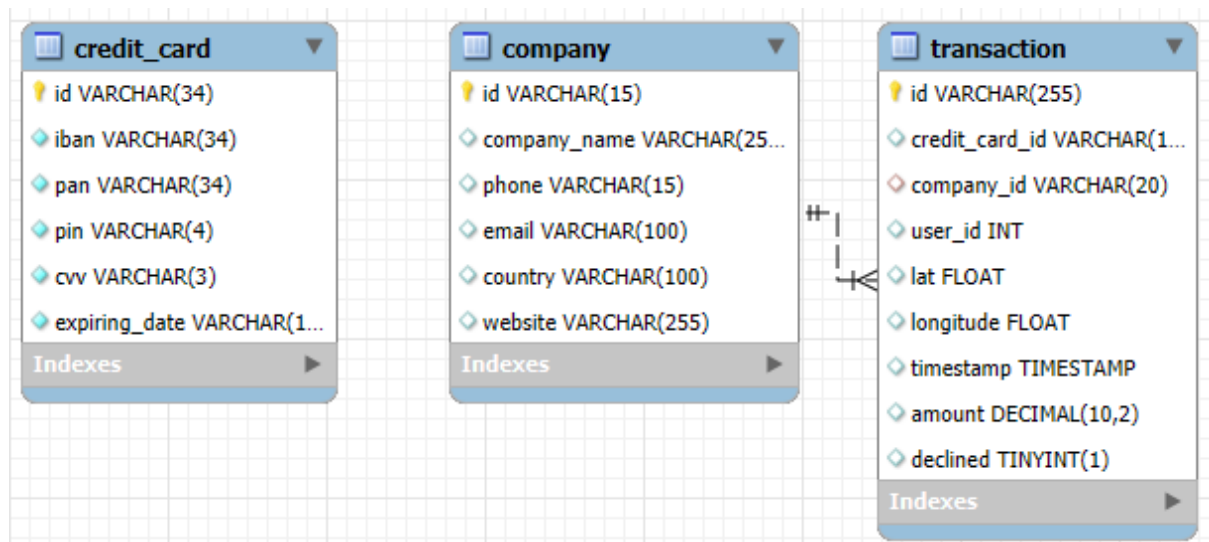
→ La tabla credit\_card tiene su estructura, pero está vacía. Por ello, hará falta introducir los datos del documento “datos\_introducir\_sprint3\_credit” en la tabla, ejecutándose:

```
Sprint3*  SQL File 7  datos_introducir_sprint3_credit x
[Icons] Don't Limit
1  -- Insertamos datos de credit_card
2  • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301954
3  • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2945', 'D026854'
4  • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2952', 'BG45IVQI
5  • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2959', 'CR72424'
6  • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2966', 'BG72LKTI

Output
Action Output
# Time Action Message Duration / Fetch
✓ 105143 13:26:24 INSERT INTO credit_card (id, iban, pan, pin, cvv... 1 row(s) affected 0.015 sec
✓ 105144 13:26:24 INSERT INTO credit_card (id, iban, pan, pin, cvv... 1 row(s) affected 0.000 sec
```

NOTA: la razón por la que el número de operaciones de la consola es muy superior al del número total de registros presente en el documento, se debe a que he vuelto a cargar la base de datos “transactions” desde cero para volver a realizar todas las operaciones de modificación de datos para el sprint 3.

→ La tabla, sin embargo, está separada de las otras dos.

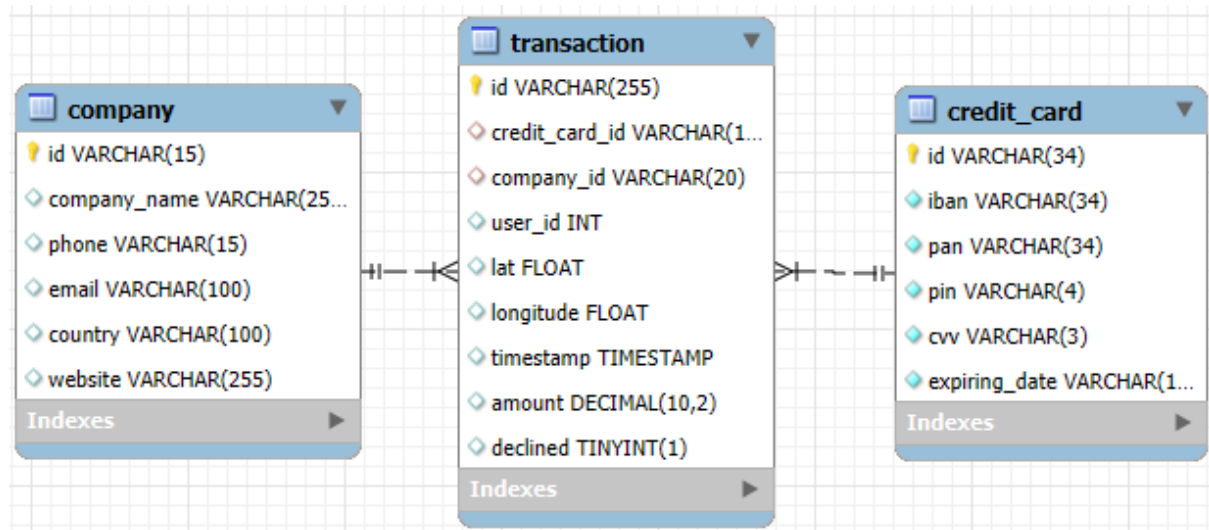


Así pues, unimos “credit\_card” a la tabla transacciones mediante una **FOREIGN KEY**. Como cada transacción va asociada o “pertenece” a una tarjeta de crédito, uniremos el campo “credit\_card\_id” de “transactions” al id (clave primaria) de “credit\_card”.

```
40 • ALTER TABLE transaction
41 ADD FOREIGN KEY (credit_card_id) REFERENCES credit_card(id);

Output
Action Output
# Time Action Message Duration / Fetch
✓ 105142 13:26:24 INSERT INTO credit_card (id, iban, pan, pin, cvv... 1 row(s) affected 0.000 sec
✓ 105143 13:26:24 INSERT INTO credit_card (id, iban, pan, pin, cvv... 1 row(s) affected 0.015 sec
✓ 105144 13:26:24 INSERT INTO credit_card (id, iban, pan, pin, cvv... 1 row(s) affected 0.000 sec
✓ 105145 13:31:34 ALTER TABLE transaction ADD FOREIGN KEY ... 100000 row(s) affected Records: 100000 Duplica... 2.079 sec
```

Este es el diagrama que queda como resultado:



Observamos las siguientes características:

- **id** - Identificador único de la tarjeta de crédito
- **loan** - Relacionado con el préstamo o línea de crédito asociada
- **pan** - Número de la tarjeta (Primary Account Number)
- **pin** - Código de seguridad personal (Personal Identification Number)
- **cvv** - Código de verificación de la tarjeta
- **expiring\_date** - Fecha de caducidad de la tarjeta.

Como ya hemos comentado, se observa que la tabla credit\_card está vinculada a la tabla maestra, "transaction", a partir del id de la targeta de crédito.

## Exercici 2

El departament de Recursos Humans ha identificat un error en el número de compte associat a la targeta de crèdit amb ID CcU-2938. La informació que ha de mostrar-se per a aquest registre és: TR323456312213576817699999. Recorda mostrar que el canvi es va realitzar.

Lo primero que deberíamos hacer es revisar que el nuevo iban no esté ya presente en la tabla credit\_card.

```
31 • SELECT * FROM credit_card
32 WHERE iban = 'TR323456312213576817699999';
```

The interface shows a 'Result Grid' with columns: id, iban, pan, pin, cvv, expiring\_date. All values are NULL. Below the grid, the 'Output' pane shows 'Action Output' with two entries:

#	Time	Action	Message	Duration / Fetch
105145	13:31:34	ALTER TABLE transaction ADD FOREIGN KE...	100000 row(s) affected Records: 100000 Duplic...	2.079 sec
105146	13:44:14	SELECT * FROM credit_card WHERE id = 'Ccu...	1 row(s) returned	0.000 sec / 0.000 sec

Vemos que no, que el IBAN no está presente en la tabla, por lo que no hay riesgo de duplicidad ni conflicto. Así pues utilizamos UPDATE en la tabla “credit\_card”, canviant el valor anterior pel nou (“TR323456312213576817699999”), en aquell camp que coincideix amb l’ID.

```
20 -- Exercici 2: Corregge el IBAN de la targeta.
21
22 • UPDATE credit_card
23     SET iban = 'TR323456312213576817699999'
24     WHERE id = 'Ccu-2938';
25 -- ara verifiquem si el canvi s'ha dut a terme correctament:
26
27 • SELECT iban
28     FROM credit_card
29     WHERE id = 'Ccu-2938'
```

The 'Result Grid' shows one row with the value TR323456312213576817699999 in the 'iban' column. The 'Output' pane shows 'Action Output' with four entries:

#	Time	Action	Message	Duration / Fetch
5047	11:23:18	SELECT * FROM credit_card -- Exer...	5000 row(s) returned	0.000 sec / 0.015 sec
5048	11:32:07	UPDATE TABLE credit_card SET ib...	Error Code: 1064. You have an error in your SQL syntax; check...	0.000 sec
5049	11:32:17	UPDATE credit_card SET iban = 'T...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.015 sec
5050	11:34:13	SELECT iban FROM credit_card W...	1 row(s) returned	0.000 sec / 0.000 sec

Després, tan sols podem fer SELECT a tota la taula i filtrar amb el buscador de la vista, o seleccionar únicament l’IBAN amb l’ID per tal de comprovar si el canvi s’ha fet correctament.

## Exercici 3

En la taula "transaction" ingresa una nova transacció amb la següent informació:

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

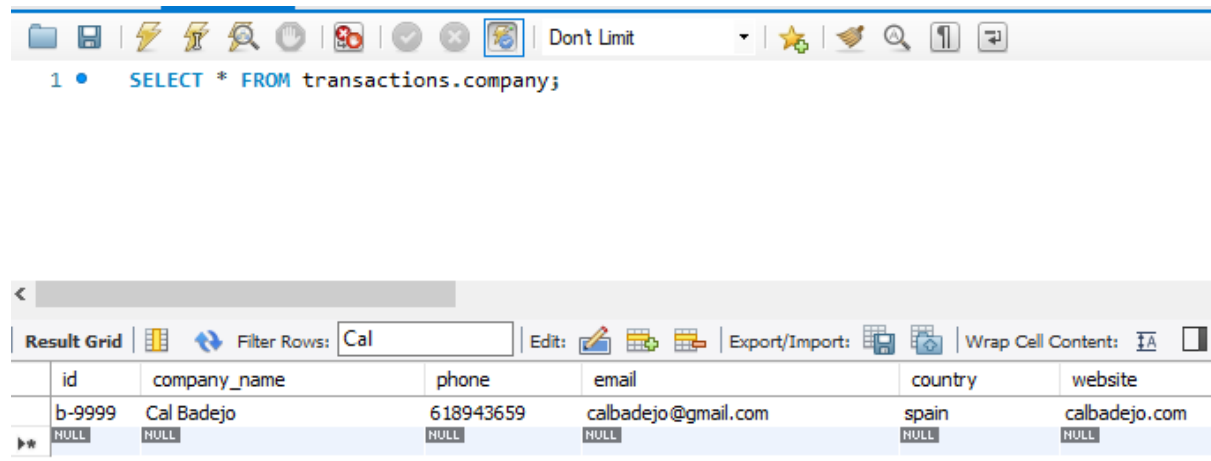
→ Si bien este ejercicio consiste en añadir un registro nuevo a la tabla transactions, dos de los campos solo están presentes en esta tabla mediante foreign keys. Por ello, no podemos insertar los datos sin antes actualizar las tablas de origen de los campos "company\_id" y "credit\_card\_id". En una situación real, tendríamos la responsabilidad de solicitar al departamento todos los datos relativos a la nueva empresa y los datos de la tarjeta del nuevo usuario, pero en este caso nos vamos a inventar el resto de los campos.

→ Así pues, primero he modificado la tabla company, usando el id solicitado:

```
34 • ○ INSERT INTO company (  
35     id,  
36     company_name,  
37     phone,  
38     email,  
39     country,  
40     website  
41 ) VALUES (  
42     'b-9999',  
43     'Cal Badejo',  
44     '618943659',  
45     'calbadejo@gmail.com',  
46     'spain',  
47     'calbadejo.com'  
48 );
```

Output					
Action Output					
#	Time	Action	Message	Duration / Fetch	
✓ 5043	14:09:30	INSERT INTO company ( id, company_nam...	1 row(s) affected	0.000 sec	
✓ 5044	14:09:38	SELECT * FROM transactions.company	101 row(s) returned	0.000 sec / 0.000 sec	
✓ 5045	14:15:05	SELECT * FROM transactions.company	101 row(s) returned	0.000 sec / 0.000 sec	
✓ 5046	14:17:28	INSERT INTO credit_card ( id, iban, pan, ...	1 row(s) affected	0.000 sec	

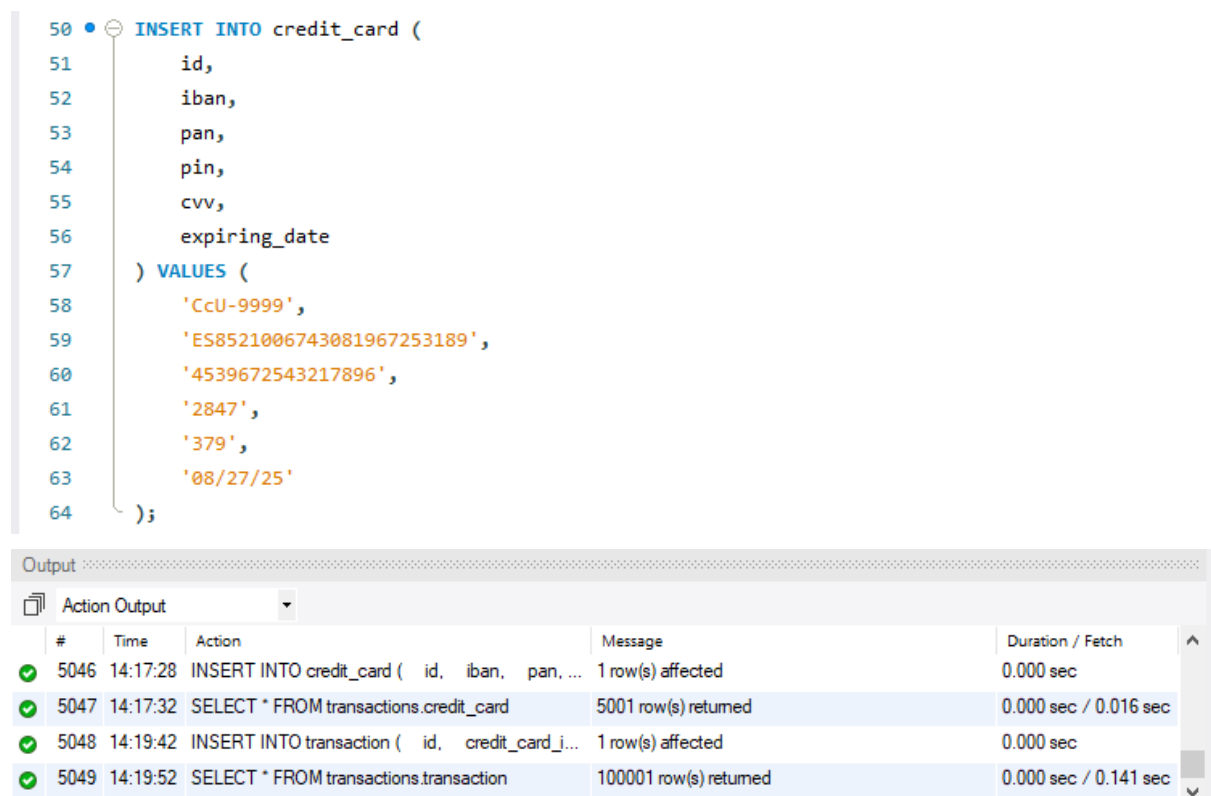
→ Comprobamos que el registro se haya llenado correctamente.



The screenshot shows a database client window with a toolbar at the top. Below the toolbar, a SQL query is entered in a text area: `1 • SELECT * FROM transactions.company;`. Below the query, a 'Result Grid' is displayed. The grid has a toolbar with options like 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The grid contains the following data:

id	company_name	phone	email	country	website
b-9999	Cal Badejo	618943659	calbadejo@gmail.com	spain	calbadejo.com
NULL	NULL	NULL	NULL	NULL	NULL

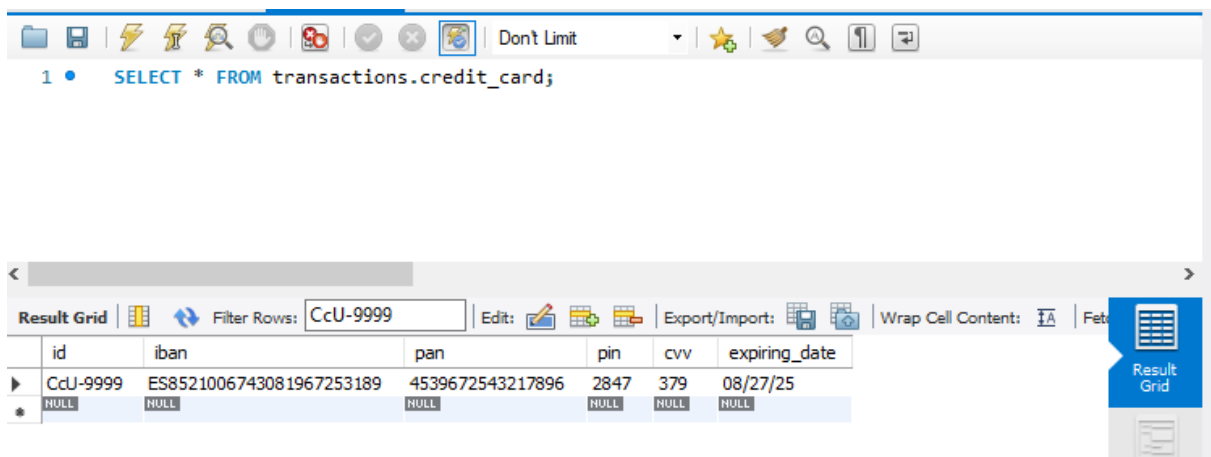
→ Luego, he añadido otro registro en la tabla credit card, usando el id solicitado



The screenshot shows a database client window with a SQL query editor. The query is: `50 • INSERT INTO credit_card ( 51 id, 52 iban, 53 pan, 54 pin, 55 cvv, 56 expiring_date 57 ) VALUES ( 58 'CcU-9999', 59 'ES8521006743081967253189', 60 '4539672543217896', 61 '2847', 62 '379', 63 '08/27/25' 64 );`. Below the query, an 'Output' window is shown with a table of execution results:

#	Time	Action	Message	Duration / Fetch
5046	14:17:28	INSERT INTO credit_card ( id, iban, pan, ...	1 row(s) affected	0.000 sec
5047	14:17:32	SELECT * FROM transactions.credit_card	5001 row(s) returned	0.000 sec / 0.016 sec
5048	14:19:42	INSERT INTO transaction ( id, credit_card_i...	1 row(s) affected	0.000 sec
5049	14:19:52	SELECT * FROM transactions.transaction	10001 row(s) returned	0.000 sec / 0.141 sec

Nuevamente, comprobamos que el registro se haya llenado correctamente.



The screenshot shows a database client window with a SQL query: `1 • SELECT * FROM transactions.credit_card;`. Below the query, a 'Result Grid' is displayed. The grid has a toolbar with options like 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. The grid contains the following data:

id	iban	pan	pin	cvv	expiring_date
CcU-9999	ES8521006743081967253189	4539672543217896	2847	379	08/27/25
NULL	NULL	NULL	NULL	NULL	NULL

```
67 • INSERT INTO transaction (  
68     id,  
69     credit_card_id,  
70     company_id,  
71     user_id,  
72     lat,  
73     longitude,  
74     amount,  
75     declined  
76 ) VALUES (  
77     '10881D1D-5B23-A76C-55EF-C568E49A99DD',  
78     'CcU-9999',  
79     'b-9999',  
80     9999,  
81     829.999,  
82     -117.999,  
83     111.11,  
84     0  
85 );
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5046	14:17:28	INSERT INTO credit_card ( id, iban, pan, ...	1 row(s) affected	0.000 sec
5047	14:17:32	SELECT * FROM transactions.credit_card	5001 row(s) returned	0.000 sec / 0.016 sec

Result Grid

Filter Rows: CcU-9999

Edit

Export/Import

Wrap Cell Content

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp
	108B1D1D-5B23-A76C-55EF-C568E49A99DD	CcU-9999	b-9999	9999	829.999	-117.999	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid



## Exercici 4

Des de recursos humans et sol·liciten eliminar la columna "pan" de la taula credit\_card. Recorda mostrar el canvi realitzat.

31 • `SELECT * FROM credit_card`

Result Grid

	id	iban	pan	pin	cvv	expiring_date
▶	CcS-4857	XX4857591835292505850771	2314242385113924	1819	467	09/27/25
	CcS-4858	XX8581768137002436094025	6582720299715533	3964	817	12/28/28
	CcS-4859	XX7826930491423553609370	8861684536289642	4983	277	11/26/26
	CcS-4860	XX5559590368835304645299	2481155515498459	6876	661	07/27/27
	CcS-4861	XX2035182877195191627307	1308930301149557	5710	398	04/25/26

credit\_card 5 x Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 105147	13:47:25	SELECT iban, COUNT(*) as cantidad FROM cr...	0 row(s) returned	0.015 sec / 0.000 sec
✓ 105148	13:48:09	SELECT * FROM credit_card WHERE iban = T...	0 row(s) returned	0.000 sec / 0.000 sec

→ La columna forma parte de la “estructura” de la tabla credit\_card, por lo que tendremos que emplear “DROP” sobre column, no “DELETE”.

```
87  -- EXERCICI 4: Eliminar la columna pan de la taula credit_card.
88
89  • ALTER TABLE credit_card
90    DROP column pan;
91
92  • SELECT * FROM credit_card
```

Result Grid

	id	iban	pin	cvv	expiring_date
▶	CcS-4857	XX4857591835292505850771	1819	467	09/27/25
	CcS-4858	XX8581768137002436094025	3964	817	12/28/28
	CcS-4859	XX7826930491423553609370	4983	277	11/26/26
	CcS-4860	XX5559590368835304645299	6876	661	07/27/27
	CcS-4861	XX2035182877195191627307	5710	398	04/25/26
	CcS-4862	XX4774721462463645409758	4042	174	11/27/26
	CcS-4863	XX1476829664245046207111	5969	449	12/27/29

credit\_card 1 x Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	10:33:42	ALTER TABLE credit_card DROP column pan	0 row(s) affected Records: 0 Duplicates: 0 Wamin...	0.047 sec
✓ 2	10:34:31	SELECT * FROM credit_card	5001 row(s) returned	0.000 sec / 0.000 sec

# Nivell 2

## Exercici 1

Elimina de la taula transaction el registre amb ID 000447FE-B650-4DCF-85DE-C7ED0EE1CAAD de la base de dades.

Antes de eliminar cualquier registro de la tabla de datos, es recomendable ver aquello que vamos a eliminar:

The screenshot shows a SQL query editor with the following SQL code:

```
100 • SELECT * FROM transaction
101 WHERE transaction.id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD'
```

Below the query, a "Result Grid" displays the results of the query. The grid has the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, and timestamp. The first row shows the record to be deleted, and the second row shows NULL values.

id	credit_card_id	company_id	user_id	lat	longitude	timestamp
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	438	41.5972	12.2218	2016-12-21 20:07:
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Below the grid, an "Output" section shows the execution log:

#	Time	Action	Message	Duration / Fetch
2	10:34:31	SELECT * FROM credit_card	5001 row(s) returned	0.000 sec / 0.000 sec
3	10:44:52	SELECT * FROM transaction WHERE transactio...	1 row(s) returned	0.000 sec / 0.000 sec

En este caso, estamos eliminando un campo de la tabla maestra, en la que están duplicados todos los datos de las demás. Por ello, eliminar el id de una transacción de esta tabla no pone en riesgo la integridad de los datos de las otras tablas con las que está vinculada. Podemos eliminar el registro sin mayor problema.

The screenshot shows a SQL query editor with the following SQL code:

```
111 • START TRANSACTION;
112 • DELETE FROM transaction
113 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
114
115 • COMMIT;
116
117 -- RECUERDA!!! FALTA PONER LA VERIFICACIÓN DE QUE ESTE ID YA NO EXISTE.
```

Below the query, an "Output" section shows the execution log:

#	Time	Action	Message	Duration / Fetch
105157	14:11:54	DELETE FROM transaction WHERE id = '00044...	0 row(s) affected	0.000 sec
105158	14:12:01	COMMIT	0 row(s) affected	0.000 sec

Sin embargo, por seguridad, decidí emplear el comando "START TRANSACTION" para hacer una "copia" de la tabla como está. De este modo, si resulta que el borrado del registro si ha afectado la integridad de los datos de alguna otra tabla, al final siempre puedo dar vuelta atrás con ROLLBACK.

Una vez me aseguré del cambio, simplemente ejecuté COMMIT. Como se observa en la imagen, el registro ya no se encuentra en la tabla.

The screenshot displays a database management interface. At the top, a SQL query is entered in a text area:

```
31 • SELECT * FROM transaction
32 WHERE id = '000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';
```

Below the query, a toolbar includes options like 'Result Grid', 'Filter Rows', 'Edit', 'Export/Import', and 'Wrap Cell Content'. A 'Result Grid' button is visible on the right. The main area shows a table with the following columns: id, credit\_card\_id, company\_id, user\_id, lat, longitude, timestamp, amount, and declined. The first row contains all NULL values.

Below the table, there is a tab labeled 'transaction 9' with 'Apply' and 'Revert' buttons. The 'Output' section is expanded, showing 'Action Output' with the following details:

#	Time	Action	Message	Duration / Fetch
✓ 105158	14:12:01	COMMIT	0 row(s) affected	0.000 sec
✓ 105159	14:13:09	DELETE FROM transaction WHERE id = '00044...	0 row(s) affected	0.000 sec

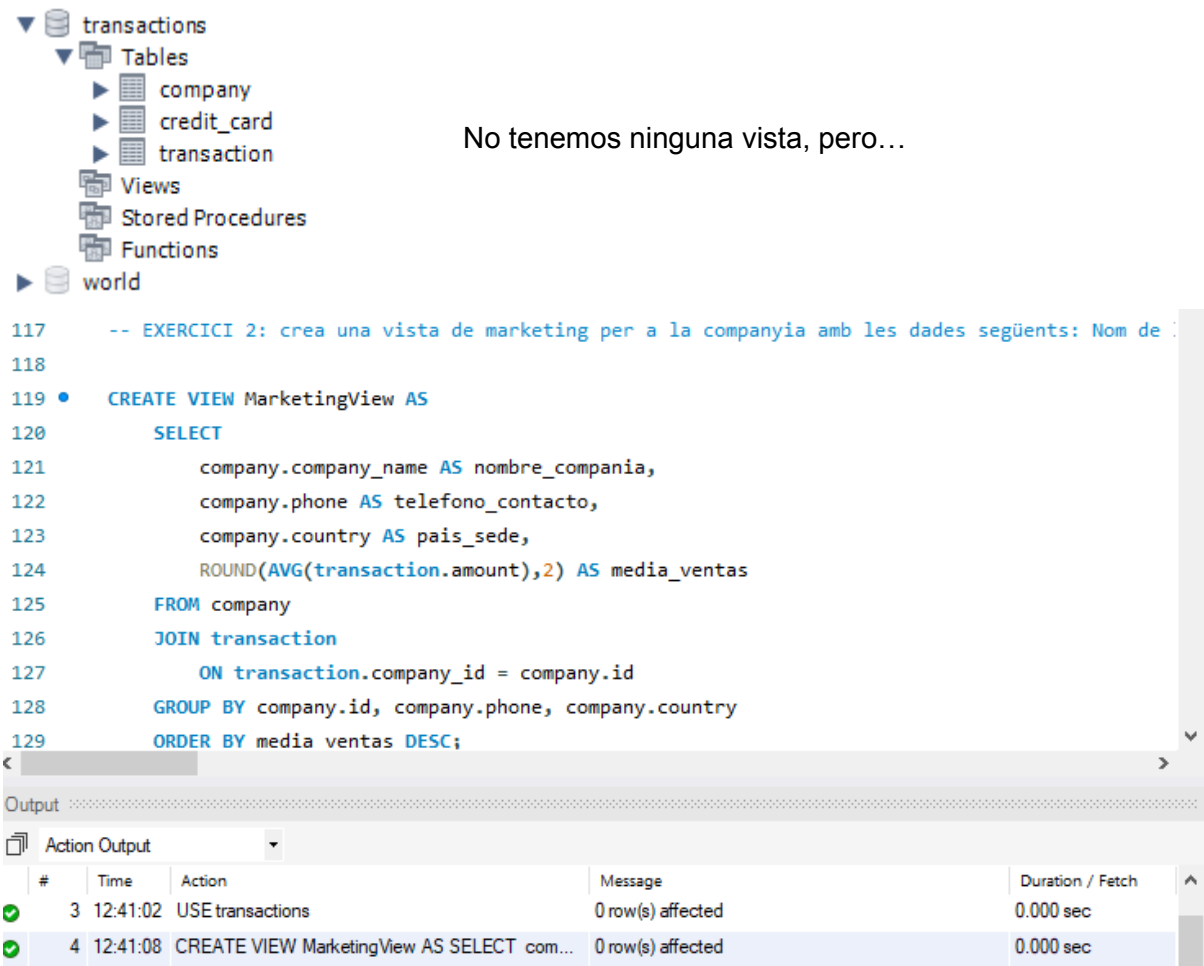
## Exercici 2

La secció de màrqueting desitja tenir accés a informació específica per a realitzar anàlisi i estratègies efectives. S'ha sol·licitat crear una vista que proporcioni detalls clau sobre les companyies i les seves transaccions. Serà necessària que creïs una vista anomenada VistaMarketing que contingui la següent informació: Nom de la companyia. Telèfon de contacte. País de residència. Mitjana de compra realitzat per cada companyia. Presenta la vista creada, ordenant les dades de major a menor mitjana de compra.

La creación de una vista es muy similar a la de una tabla, solo que aquí juega un papel importante JOIN si tomamos campos de varias tablas, como en una consulta normal; ya que los datos que necesitamos ya están insertados.

```
CREATE VIEW MarketingView AS
SELECT
    company.company_name AS nombre_compania,
    company.phone AS telefono_contacto,
    company.country AS pais_sede,
    ROUND(AVG(transaction.amount),2) AS media_ventas
FROM company
JOIN transaction
    ON transaction.company_id = company.id
GROUP BY company.id, company.phone, company.country
ORDER BY media_ventas DESC;
```

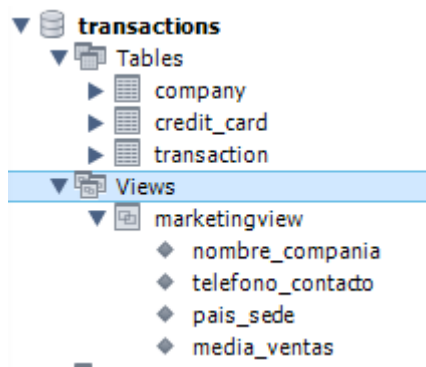
No tenemos ninguna vista, pero...



The screenshot shows a database IDE with a tree view on the left containing 'transactions', 'Tables' (with 'company', 'credit\_card', 'transaction'), 'Views', 'Stored Procedures', 'Functions', and 'world'. The main editor displays the SQL code for creating the 'MarketingView' view. Below the editor is an 'Output' window showing the execution results.

```
117 -- EXERCICI 2: crea una vista de marketing per a la companyia amb les dades següents: Nom de :
118
119 • CREATE VIEW MarketingView AS
120     SELECT
121         company.company_name AS nombre_compania,
122         company.phone AS telefono_contacto,
123         company.country AS pais_sede,
124         ROUND(AVG(transaction.amount),2) AS media_ventas
125     FROM company
126     JOIN transaction
127         ON transaction.company_id = company.id
128     GROUP BY company.id, company.phone, company.country
129     ORDER BY media_ventas DESC;
```

#	Time	Action	Message	Duration / Fetch
✓ 3	12:41:02	USE transactions	0 row(s) affected	0.000 sec
✓ 4	12:41:08	CREATE VIEW MarketingView AS SELECT com...	0 row(s) affected	0.000 sec



Esta vista combina datos muy útiles para una campaña de marketing, y permite consultar rápidamente el conjunto de datos en específico que estamos buscando, en lugar de tener que reprogramar todo nuevamente.

32 • `SELECT * FROM transactions.marketingview;`

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

nombre_compania	telefono_contacto	pais_sede	media_ventas
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Pretium Neque Corp.	07 77 48 55 28	Australia	276.16
Urna Convallis Associates	06 01 24 77 04	United States	274.24
At Associates	09 56 61 10 65	New Zealand	272.21
Metus Vitae Associates	08 25 44 40 66	Australia	270.08

marketingview 1 x | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
31	13:11:16	select * from company	101 row(s) returned	0.000 sec / 0.000 sec
32	13:22:26	SELECT * FROM transactions.marketingview	101 row(s) returned	0.407 sec / 0.000 sec

## Exercici 3

Filtra la vista VistaMarketing per a mostrar només les companyies que tenen el seu país de residència en "Germany".

136 • `SELECT *`  
 137 `FROM marketingview`  
 138 `WHERE pais_sede = 'Germany';`

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

nombre_compania	telefono_contacto	pais_sede	media_ventas
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Nunc Interdum Incorporated	05 18 15 48 13	Germany	259.32
Convallis In Incorporated	06 66 57 29 50	Germany	257.75
Ac Industries	09 34 65 40 60	Germany	255.15
Rutrum Non Inc.	02 66 31 61 09	Germany	255.14
Auctor Mauris Corp.	05 62 87 14 41	Germany	254.77

marketingview 1 x | Read Only

Output

Action Output

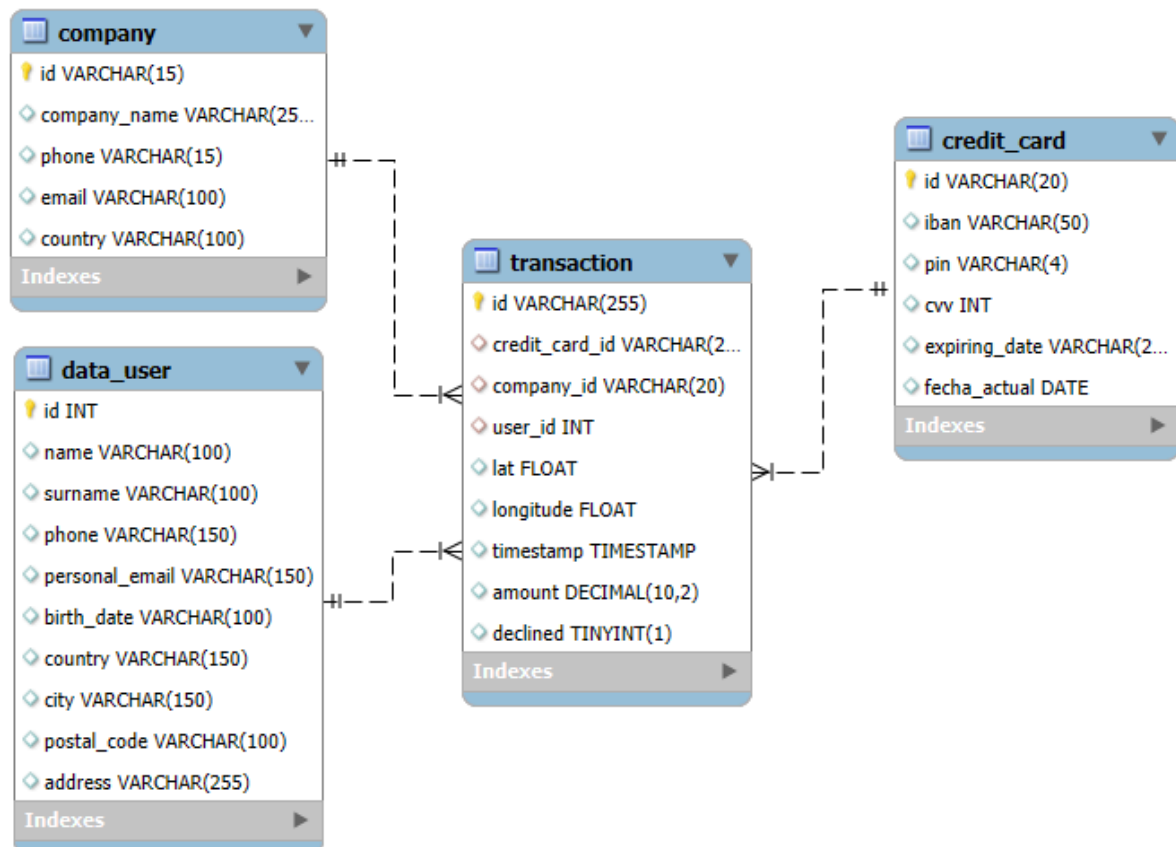
#	Time	Action	Message	Duration / Fetch
32	13:22:26	SELECT * FROM transactions.marketingview	101 row(s) returned	0.407 sec / 0.000 sec
33	13:24:13	SELECT * FROM marketingview WHERE pais_s...	8 row(s) returned	0.094 sec / 0.000 sec

→ Podemos filtrar, bien mediante código, o bien con el buscador integrado en la vista.

# Nivell 3

## Exercici 1

La setmana vinent tindràs una nova reunió amb els gerents de màrqueting. Un company del teu equip va realitzar modificacions en la base de dades, però no recorda com les va realitzar. Et demana que l'ajudis a deixar els comandos executats per a obtenir el següent diagrama:



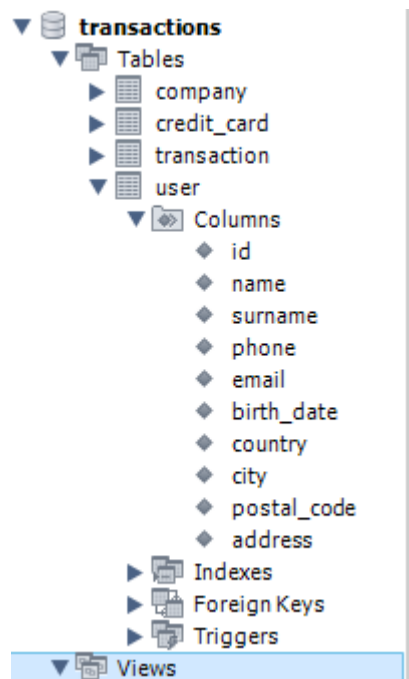
- 1) Primero, tendrás que crear la estructura de la tabla (presente en el documento "estructura\_dades\_user" que debemos importar).

```
29 • CREATE TABLE IF NOT EXISTS user (
30     id CHAR(10) PRIMARY KEY,
31     name VARCHAR(100),
32     surname VARCHAR(100),
33     phone VARCHAR(150),
34     email VARCHAR(150),
35     birth_date VARCHAR(100),
36     country VARCHAR(150),
37     city VARCHAR(150),
38     postal_code VARCHAR(100),
39     address VARCHAR(255)
40 );
```

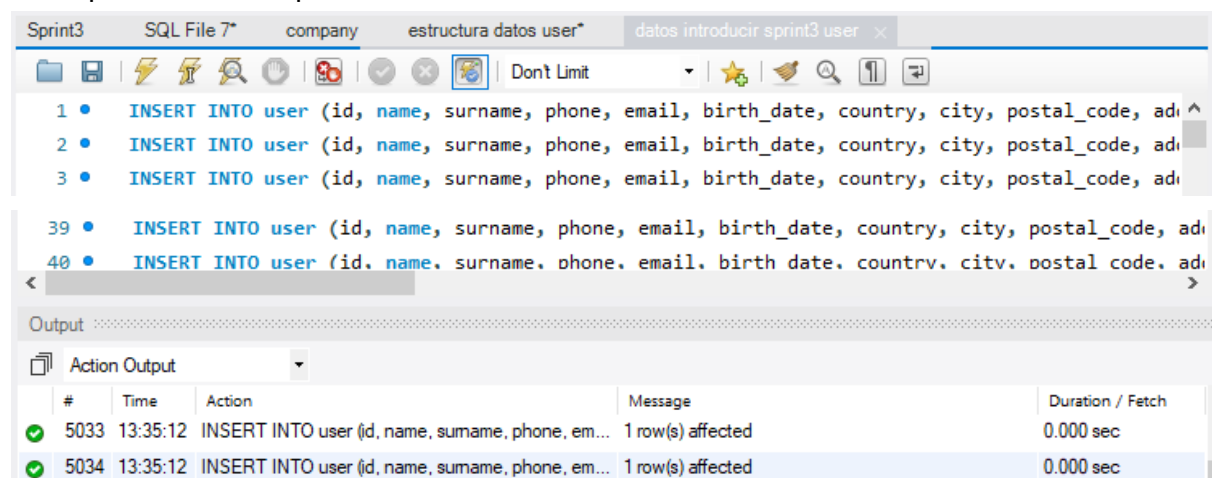
Output

#	Time	Action	Message	Duration / Fetch
33	13:24:13	SELECT * FROM marketingview WHERE pais_s...	8 row(s) returned	0.094 sec / 0.000 sec
34	13:29:13	CREATE TABLE IF NOT EXISTS user (id CHAR...	0 row(s) affected	0.016 sec

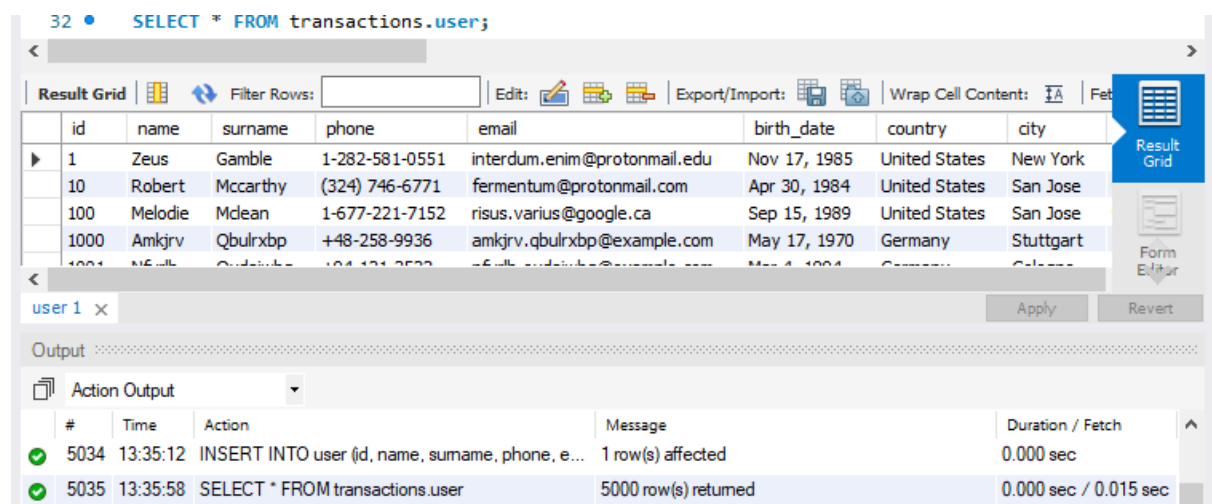
→ Podemos verificar fácilmente en el navegador lateral que la estructura de la tabla se ha creado:



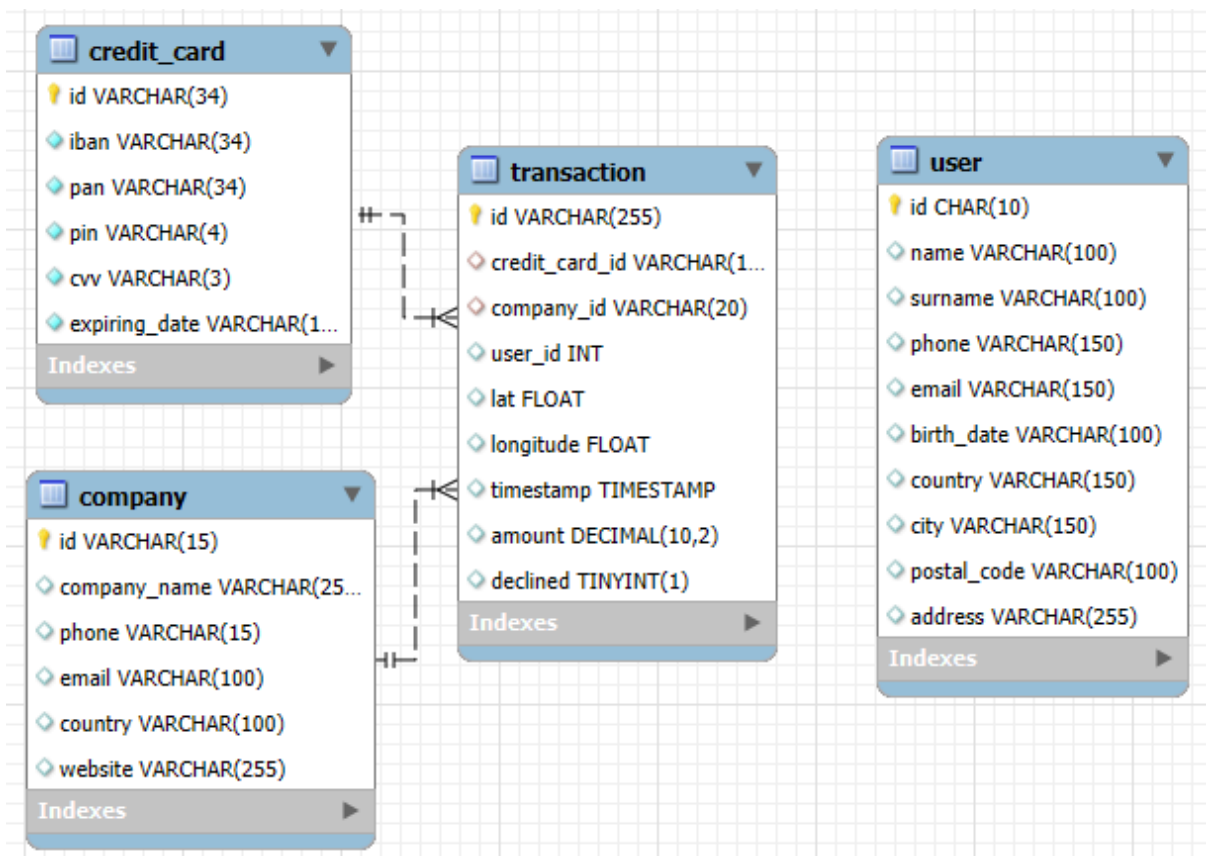
2) Ahora importamos el documento “datos\_introducir\_sprint3\_user.sql” para insertar en los correspondientes campos sus datos.



Entonces, verificamos que los datos se hayan insertado correctamente en la tabla:



Por último, si ejecutamos el diagrama, nos quedará esto:



Aquí nos encontramos ante el problema de que la tabla users se encuentra desvinculada de la tabla transactions. Por ello, tendremos que unirlas mediante una FOREIGN KEY, pero nos encontramos con dos problemas:

1. En el nivel 1, hace rato, hemos insertado en la tabla transaction un registro nuevo que no existe en la tabla user.
2. El user.id de la tabla user es un tipo de dato diferente al user\_id y la FK no puede unir datos incompatibles.

Así pues, empezamos añadiendo el registro faltante a la tabla user, solicitando al departamento de atención al usuario los datos faltantes de ese usuario faltante:

```

158 • INSERT INTO user ( id, name, surname, phone, email, birth_date, country, city, postal_code, i
159       ) VALUES ( '9999', 'Zeus', 'Gamble', '1-282-581-0551', 'interdum.enim@protonmail.edu',
160     );
161 • SELECT* FROM user
162   WHERE id = '9999';
  
```

id	name	surname	phone	email	birth_date	country	city	post
9999	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York	1000

user 2 x [Apply] [Revert]

Output

#	Time	Action	Message	Duration / Fetch
5036	13:47:01	INSERT INTO user (id, name, surname, phone, ...	1 row(s) affected	0.000 sec
5037	13:47:47	SELECT* FROM user WHERE id = '9999' -- Exist...	Error Code: 1064. You have an error in your SQL ...	0.000 sec
5038	13:47:59	SELECT* FROM user WHERE id = '9999'	1 row(s) returned	0.000 sec / 0.000 sec



→ Después revisamos con SELECT que el registro entero esté presente en la tabla por el id.

En cuanto al segundo problema, he utilizado el comando DESCRIBE user para revisar tanto el tipo de datos de la tabla user, como las FK presentes, confirmando que está completamente desvinculada al resto.

166 • DESCRIBE user; -- La tabla nos muestra la estructura de la tabla user, incluidas los tipos de

167

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

Field	Type	Null	Key	Default	Extra
id	char(10)	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 3 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5037	13:47:47	SELECT* FROM user WHERE id = '9999' -- Exist...	Error Code: 1064. You have an error in your SQL ...	0.000 sec
5038	13:47:59	SELECT* FROM user WHERE id = '9999'	1 row(s) returned	0.000 sec / 0.000 sec
5039	13:52:55	DESCRIBE user	10 row(s) returned	0.000 sec / 0.000 sec

Así decidí cambiar su clave primaria (id) a INT para hacerlo compatible con user\_id de transaction.

168 • ALTER TABLE user

169 MODIFY COLUMN id INT; -- Modificamos el tipo de dato de user.id.

170

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

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
surname	varchar(100)	YES		NULL	
phone	varchar(150)	YES		NULL	
email	varchar(150)	YES		NULL	
birth_date	varchar(100)	YES		NULL	
country	varchar(150)	YES		NULL	
city	varchar(150)	YES		NULL	
postal_code	varchar(100)	YES		NULL	
address	varchar(255)	YES		NULL	

Result 4 x Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5039	13:52:55	DESCRIBE user	10 row(s) returned	0.000 sec / 0.000 sec
5040	13:54:09	ALTER TABLE user MODIFY COLUMN id INT	5001 row(s) affected Records: 5001 Duplicates: ...	0.141 sec
5041	13:54:18	DESCRIBE user	10 row(s) returned	0.000 sec / 0.000 sec

Ahora si, podemos vincular la tabla user a transaction mediante una foreign key, que he denominado como “fk\_transaction\_user” para crear una nueva relación 1 a muchos (1:N) entre ambas tablas y para poderla identificar más fácilmente.

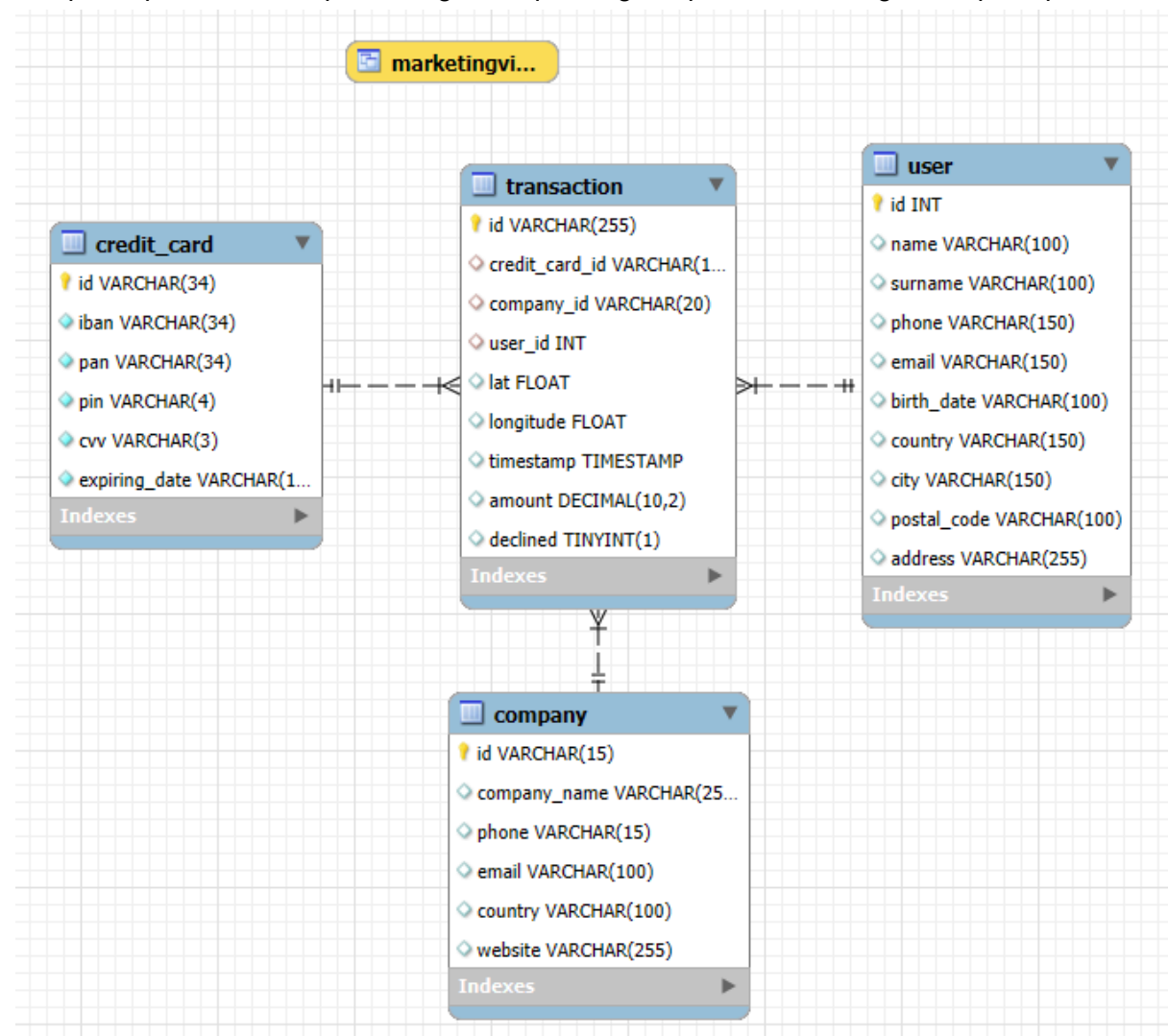
```
173 • ALTER TABLE transaction
174     ADD CONSTRAINT fk_transaction_user
175     FOREIGN KEY (user_id) REFERENCES user(id)
176     ON DELETE RESTRICT
177     ON UPDATE CASCADE;
178
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 5040	13:54:09	ALTER TABLE user MODIFY COLUMN id INT	5001 row(s) affected Records: 5001 Duplicates: ...	0.141 sec
✓ 5041	13:54:18	DESCRIBE user	10 row(s) returned	0.000 sec / 0.000 sec
✓ 5042	13:57:23	ALTER TABLE transaction ADD CONSTRAINT f...	100000 row(s) affected Records: 100000 Duplic...	2.562 sec

Así, pues, podemos ver que el diagrama queda igual que el de la imagen del principio:

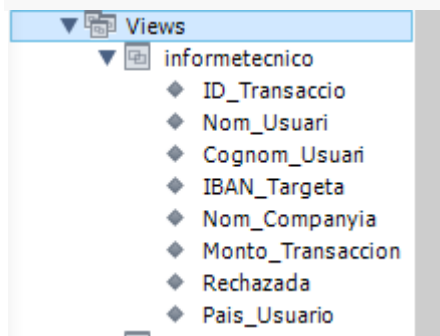


## Exercici 2

L'empresa també us demana crear una vista anomenada "InformeTecnico" que contingui la següent informació:

- ID de la transacció
- Nom de l'usuari/ària
- Cognom de l'usuari/ària
- IBAN de la targeta de crèdit usada.
- Nom de la companyia de la transacció realitzada.
- Assegureu-vos d'incloure informació rellevant de les taules que coneixereu i utilitzeu àlies per canviar de nom columnes segons calgui.

Mostra els resultats de la vista, ordena els resultats de forma descendent en funció de la variable ID de transacció.



Para esta vista he añadido estos 3 campos adicionales:

- Total de transacciones
- El estado de las transacciones
- El país del usuario

Has seleccionado indicadores cruciales para convertir el InformeTecnico en una herramienta de diagnóstico y análisis de riesgo, dirigida a los equipos de Operaciones, Soporte Técnico y Detección de Fraude. Al incluir el Monto de la Transacción, permites a los técnicos priorizar la resolución de incidentes basándose en el impacto financiero. El campo Rechazada (declined) ofrece un diagnóstico inmediato sobre el éxito o fracaso de la operación, esencial para la eficiencia del soporte. Finalmente, el País del Usuario dota al informe del contexto geográfico necesario para identificar patrones de riesgo, analizar el cumplimiento regulatorio o detectar problemas de servicio localizados.

Resultado:

```

183 • CREATE OR REPLACE VIEW InformeTecnico AS -- Usamos OR REPLACE para actualizar la vista
184 SELECT
185     transaction.id AS 'ID_Transaccio',
186     user.name AS 'Nom_Usuari',
187     user.surname AS 'Cognom_Usuari',
188     credit_card.iban AS 'IBAN_Targeta',
189     company.company_name AS 'Nom_Companyia',
190     transaction.amount AS 'Muntant_transacció',    -- 1. Importe de la transacción
191     transaction.declined AS 'Rebutjada',           -- 2. Estado (0 = Éxito, 1 = Fallo)
192     user.country AS 'País_Usuari'                  -- 3. País de origen del usuario
193 FROM
194     transaction
195 JOIN
196     user ON transaction.user_id = user.id
197 JOIN
198     credit_card ON transaction.credit_card_id = credit_card.id
199 JOIN
200     company ON transaction.company_id = company.id
201 ORDER BY transaction.id DESC;
202
203 • SELECT * FROM transactions.informetecnico;
204

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	ID_Transaccio	Nom_Usuari	Cognom_Usuari	IBAN_Targeta	Nom_Compa
▶	FFFD31D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgli	Tprvvmrc	XX794814451211289182490922	Turpis Compa
	FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfrled	Vilqcdl	XX636251701647892036676034	Amet Nulla Dr
	FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	XX162677143304223631437567	Nunc Interdu
	FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzjpa	Uirzjulh	XX395114267082019952567052	Viverra Done
	FFF9E3CE-234E-408C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537570367941	Convallis In I
	FFF9E178-6CD2-4DF9-99B0-49AE068809B1	Jevepx	Xwcwzwnm	XX321405515711654384711481	Mus Aenean I
	FFF867C9-17B5-4B1F-AFD9-F8023AAA449E	Fqlngd	Lvhfqyxi	XX278446342932680979729426	Cras Vehicula
	FFF7042D-18C6-4DDD-823C-4D90A4AC8F26	Njoraa	Egsqcuii	XX405009272572550082027209	Placerat LLP
	FFF660D4-4244-47F6-9210-E5D1DCB99DB0	Lopzaj	Itgryfay	XX63376659736627454015125	Pede Cum Ltc

informetecnico 5 x

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 5042	13:57:23	ALTER TABLE transaction ADD CONSTRAINT f...	100000 row(s) affected Records: 100000 Duplic...	2.562 sec
✓ 5043	14:01:20	CREATE VIEW InformeTecnico AS SELECT t...	0 row(s) affected	0.000 sec
✓ 5044	14:05:53	CREATE OR REPLACE VIEW InformeTecnico A...	0 row(s) affected	0.000 sec
✓ 5045	14:08:05	SELECT * FROM transactions.informetecnico	100000 row(s) returned	0.812 sec / 0.063 sec
✓ 5046	14:09:05	SELECT * FROM transactions.informetecnico	100000 row(s) returned	1.000 sec / 0.046 sec