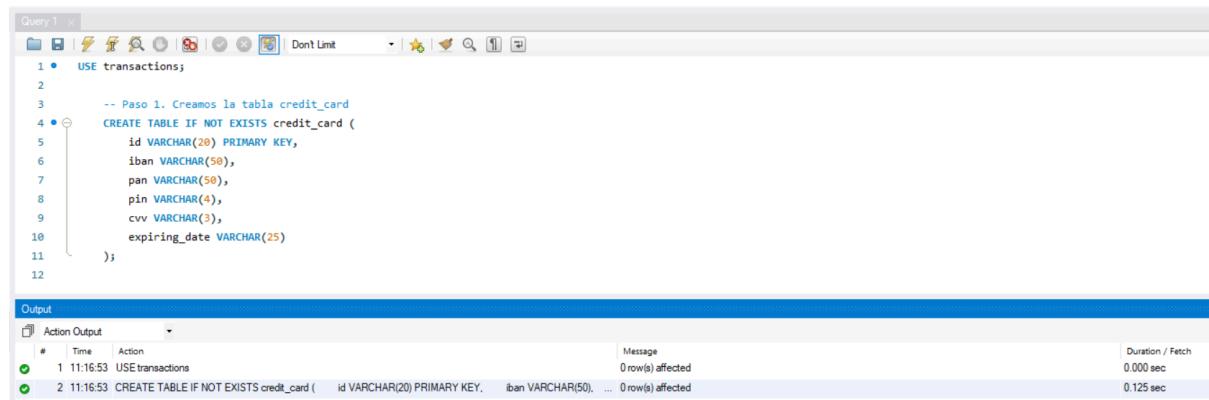


# Nivel 1

## 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.



The screenshot shows the MySQL Workbench interface with a query editor and an output window. The query editor contains the following SQL code:

```
Query 1
1 • USE transactions;
2
3 -- Paso 1. Creamos la tabla credit_card
4 • CREATE TABLE IF NOT EXISTS credit_card (
5     id VARCHAR(20) PRIMARY KEY,
6     iban VARCHAR(50),
7     pan VARCHAR(50),
8     pin VARCHAR(4),
9     cvv VARCHAR(3),
10    expiring_date VARCHAR(25)
11 );
12
```

The output window shows the results of the execution:

Action	Time	Action	Message	Duration / Fetch
1	11:16:53	USE transactions	0 row(s) affected	0.000 sec
2	11:16:53	CREATE TABLE IF NOT EXISTS credit_card ( id VARCHAR(20) PRIMARY KEY, iban VARCHAR(50), ...	0 row(s) affected	0.125 sec

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.

```
-- Paso 2. Insertamos datos de credit_card
1 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984', '10/30/22');
2 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2945', 'D026854763748537475216568689', '5142423821948828', '9880', '887', '08/24/23');
3 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2952', '8645IVQL52710525608255', '4556 453 55 5287', '4598', '438', '06/29/21');
4 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2959', 'CR7242477244335841535', '372461377349375', '3583', '667', '02/24/23');
5 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2966', 'BG72LKTQ15627628377363', '448566 886747 7265', '4900', '130', '10/29/24');
6 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2973', 'PT87806228135092429456346', '544 58654 54343 384', '8760', '887', '01/30/25');
7 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2980', 'DE39241881883086277136', '482400 7145845969', '5075', '596', '07/24/22');
8 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2987', 'GE89681434837748781813', '3763 747687 76666', '2298', '797', '10/31/23');
9 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-2994', 'BH62714428368066765294', '344283273252593', '7545', '595', '02/28/22');
10 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3001', 'CY49087426654774581266832110', '511722 924833 2244', '9562', '867', '09/16/22');
11 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3008', 'LU507216693616119230', '4485744464433884', '1856', '740', '04/05/25');
12 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3015', 'P5119398216295715968342456821', '3784 662233 17389', '3246', '822', '01/31/22');
13 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3022', 'GT9169516285056977423121857', '5164 1379 4842 3951', '5610', '342', '04/25/25');
14 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3029', 'AZ6231741398244118123739746', '3429 279566 77631', '9708', '505', '09/02/23');
15 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3036', 'AZ39336002925842865843941994', '3768 451556 48766', '2232', '565', '10/27/25');
16 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3043', 'TN6488143310514852179535', '455676 6437463635', '5969', '196', '06/07/25');
17 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('Ccu-3043')
```

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.

```
-- Paso 3. Creamos la relación con la tabla transaction
1 • ALTER TABLE transaction
2 • ADD CONSTRAINT fk_transaction_credit_card
3 • FOREIGN KEY (credit_card_id) REFERENCES credit_card(id); |
```

El diagrama representa la relación entre 3 tablas: company, transaction y credit\_card.

- “company” almacena la información de cada empresa (id, nombre, teléfono, 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:

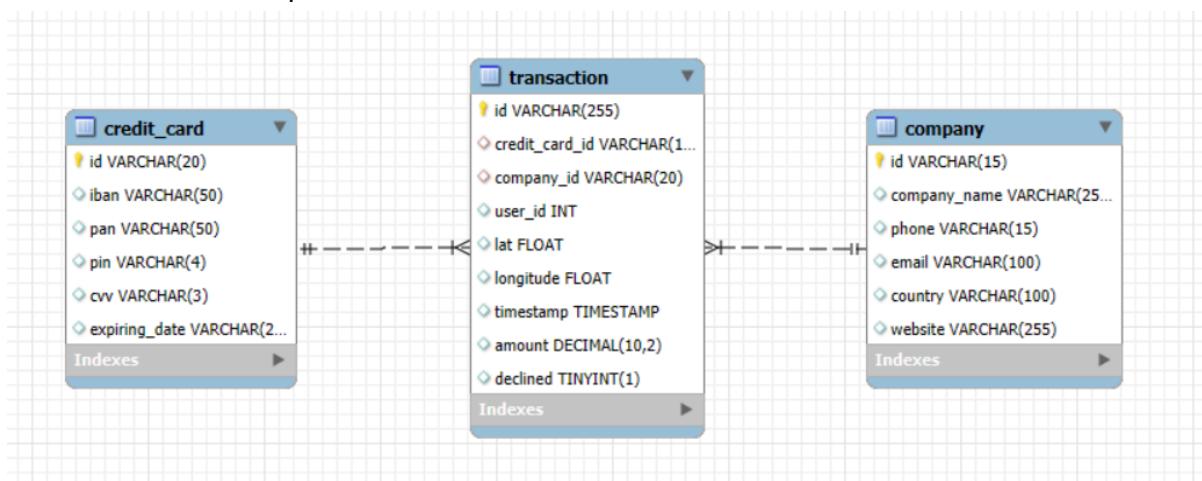
- company\_id, que indica qué empresa realizó la transacción.
- credit\_card\_id, que indica con qué tarjeta se efectuó el pago.

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

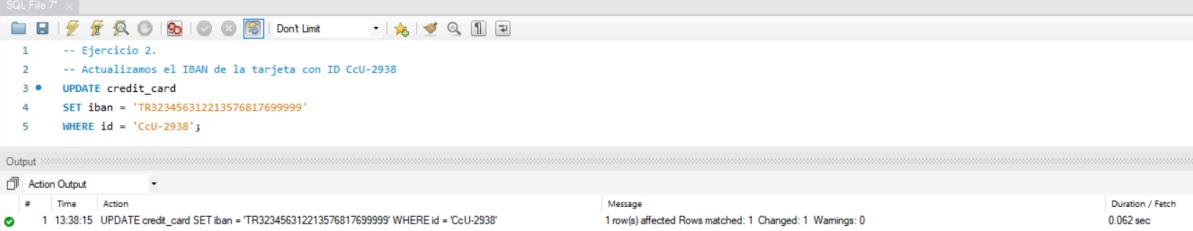
Es decir, una empresa puede tener muchas transacciones, y una tarjeta también puede estar asociada a múltiples transacciones.



## 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.

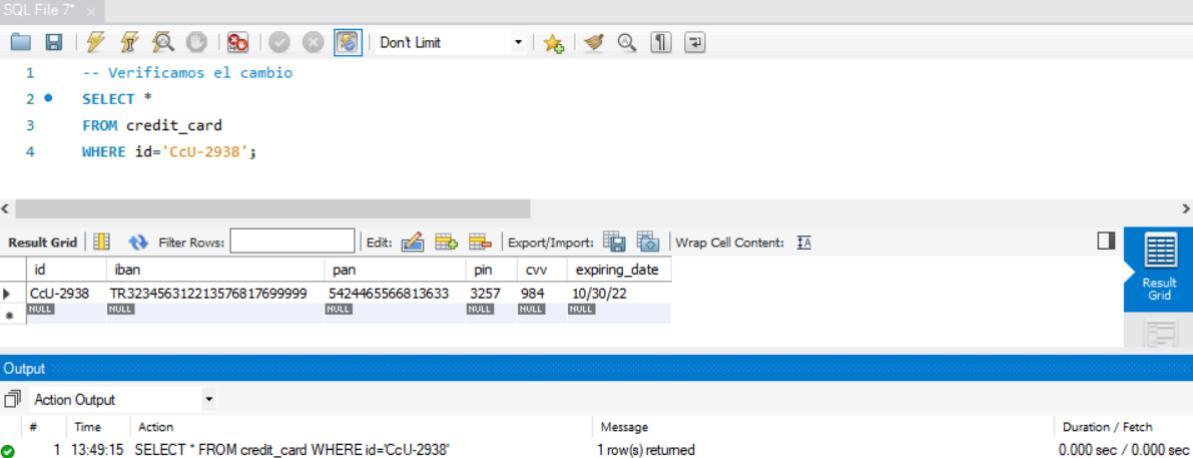
Utilizamos UPDATE para modificar la tabla “credit\_card”, con el SET asignamos el nuevo valor al campo IBAN gracias a WHERE para una única fila con el id = 'CcU-2938', si se omite WHERE, se actualizan todas las filas.



```
SQL File 7* ×
File | New | Open | Save | Import | Export | Help | Don't Limit | 
1 -- Ejercicio 2.
2 -- Actualizamos el IBAN de la tarjeta con ID CcU-2938
3 • UPDATE credit_card
4   SET iban = 'TR323456312213576817699999'
5   WHERE id = 'CcU-2938';

Output
Action Output
# Time Action
1 13:38:15 UPDATE credit_card SET iban = 'TR323456312213576817699999' WHERE id = 'CcU-2938'
Message
1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
Duration / Fetch
0.062 sec
```

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



```
SQL File 7* ×
File | New | Open | Save | Import | Export | Help | Don't Limit | 
1 -- Verificamos el cambio
2 • SELECT *
3   FROM credit_card
4   WHERE id='CcU-2938';

Result Grid | Filter Rows: [ ] | Edit: [ ] | Export/Import: [ ] | Wrap Cell Content: [ ]
Result Grid
id iban pan pin cvv expiring_date
CcU-2938 TR323456312213576817699999 5424465566813633 3257 984 10/30/22
HULL HULL HULL NULL NULL NULL

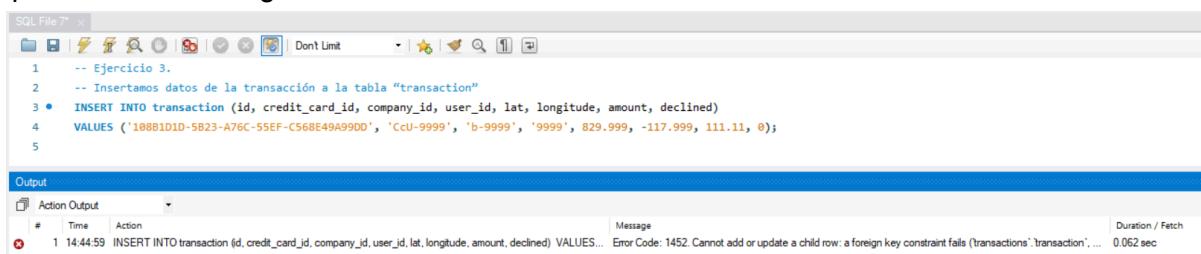
Output
Action Output
# Time Action
1 13:49:15 SELECT * FROM credit_card WHERE id='CcU-2938'
Message
1 row(s) returned
Duration / Fetch
0.000 sec / 0.000 sec
```

## 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.



The screenshot shows the MySQL Workbench interface. In the SQL editor tab, the following SQL code is written:

```
1 -- Ejercicio 3.
2 -- Insertamos datos de la transacción a la tabla "transaction"
3 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
4 VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);
```

In the Output tab, the results of the execution are shown:

#	Time	Action	Message	Duration / Fetch
1	14:44:59	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)	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('transactions.transaction', ...)	0.062 sec

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":

The screenshot shows the SQL Developer interface with a query window titled "SQL File 7". The query is:

```
1 •  SELECT * FROM credit_card WHERE id = 'CcU-9999';
```

The result grid shows the following columns: id, iban, pan, pin, cvv, and expiring\_date. All values are null.

The output pane shows the following log entry:

#	Time	Action	Message	Duration / Fetch
1	14:12:00	SELECT * FROM credit_card WHERE id = 'CcU-9999'	0 row(s) returned	0.000 sec / 0.000 sec

- company\_id = 'b-9999' no existe en la tabla "company":

The screenshot shows the SQL Developer interface with a query window titled "SQL File 7". The query is:

```
1 •  SELECT * FROM company WHERE id = 'b-9999';
```

The result grid shows the following columns: id, company\_name, phone, email, country, and website. All values are null.

The output pane shows the following log entry:

#	Time	Action	Message	Duration / Fetch
1	14:13:24	SELECT * FROM company WHERE id = b-9999'	0 row(s) returned	0.047 sec / 0.000 sec

Para solucionarlo, primero se crean los registros padre en credit\_card y company.

The screenshot shows two separate SQL developer sessions.

The top session shows the following query:

```
1 •  INSERT INTO credit_card(id)
2   VALUES ('CcU-9999');
```

The output shows the insert was successful:

#	Time	Action	Message	Duration / Fetch
1	14:27:48	INSERT INTO credit_card(id) VALUES (CcU-9999)	1 row(s) affected	0.063 sec

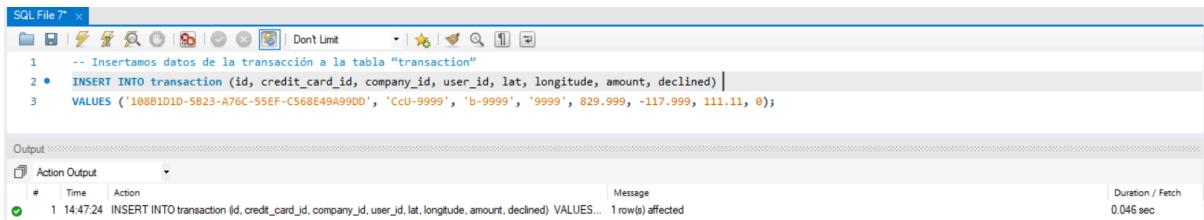
The bottom session shows the following query:

```
1 •  INSERT INTO company(id)
2   VALUES ('b-9999');
```

The output shows the insert was successful:

#	Time	Action	Message	Duration / Fetch
1	14:30:36	INSERT INTO company(id) VALUES (b-9999)	1 row(s) affected	0.047 sec

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



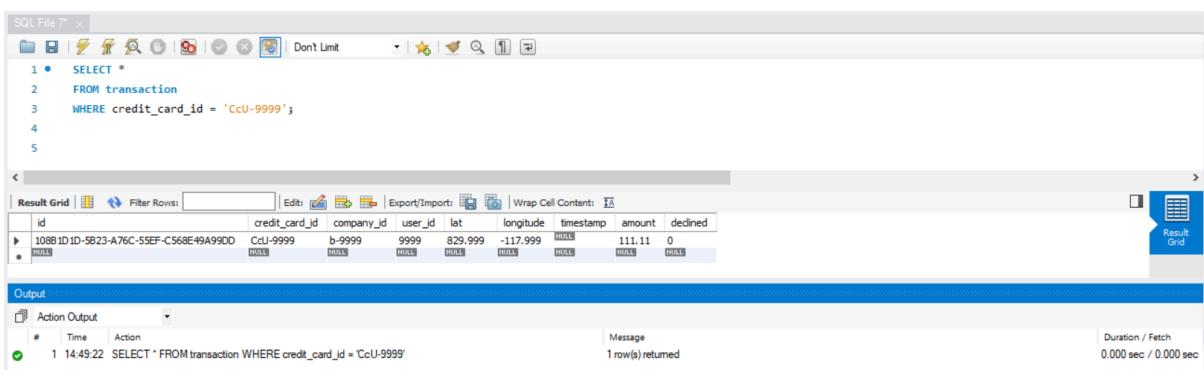
SQL File 7

```
1 -- Insertamos datos de la transacción a la tabla "transaction"
2 • INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined)
3 VALUES ('10881D1D-5823-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', 829.999, -117.999, 111.11, 0);
```

Output

#	Time	Action	Message	Duration / Fetch
1	14:47:24	INSERT INTO transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) VALUES...	1 row(s) affected	0.046 sec

Lo comprobamos:



SQL File 7

```
1 • SELECT *
2 FROM transaction
3 WHERE credit_card_id = 'CcU-9999';
4
5
```

Result Grid

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
10881D1D-5823-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

Output

#	Time	Action	Message	Duration / Fetch
1	14:49:22	SELECT * FROM transaction WHERE credit_card_id = 'CcU-9999'	1 row(s) returned	0.000 sec / 0.000 sec

## Ejercicio 4

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

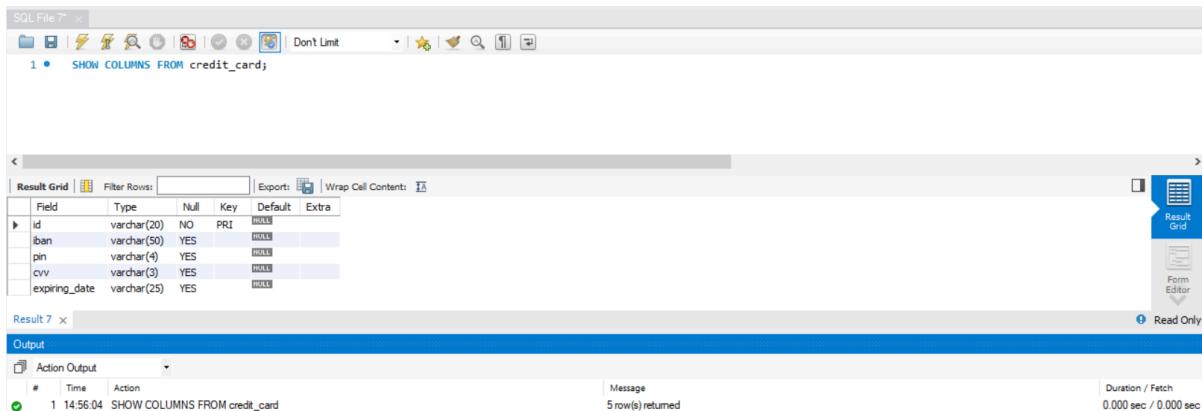
Modificamos la estructura de la tabla con ALTER TABLE, para eliminar la columna utilizamos DROP COLUMN.



```
SQL File 7* ×
File Edit View Insert Object Tools Help
1 -- Ejercicio 4.
2 • ALTER TABLE credit_card
3   DROP COLUMN pan;
4

Output
Action Output
# Time Action
1 14:53:11 ALTER TABLE credit_card DROP COLUMN pan
Message
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
Duration / Fetch
0.062 sec
```

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



```
SQL File 7* ×
File Edit View Insert Object Tools Help
1 • SHOW COLUMNS FROM credit_card;

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
Field Type Null Key Default Extra
id varchar(20) NO PRI NULL
iban varchar(50) YES NULL
pin varchar(4) YES NULL
cvv varchar(3) YES NULL
expiring_date varchar(25) YES NULL

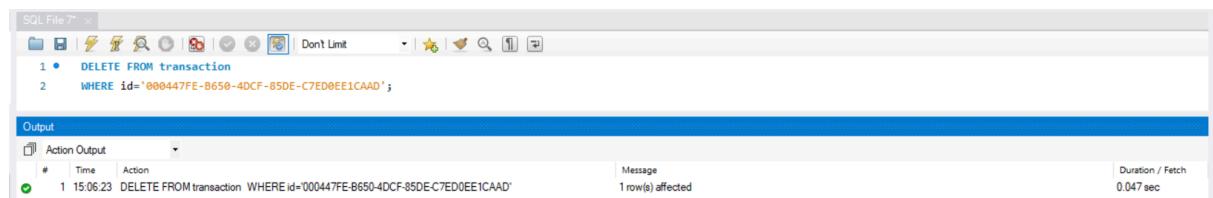
Result 7 ×
Output
Action Output
# Time Action
1 14:56:04 SHOW COLUMNS FROM credit_card
Message
5 row(s) returned
Duration / Fetch
0.000 sec / 0.000 sec
```

# 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.



The screenshot shows a SQL File window in SSMS with the following content:

```
SQL File 7" x
[File] [New] [Save] [Save All] [Close] [Find] [Replace] [Copy] [Cut] [Delete] [Select] [Dont Limit] | [Star] [Zoom] [Search] [Help] [Exit]
1 • DELETE FROM transaction
2 WHERE id='000447FE-B650-4DCF-85DE-C7ED0EE1CAAD';

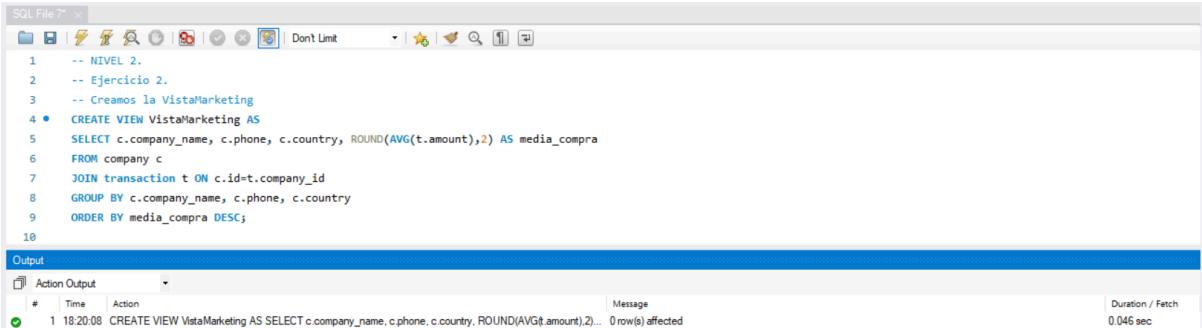
Output
Action Output
# Time Action Message Duration / Fetch
1 15:06:23 DELETE FROM transaction WHERE id='000447FE-B650-4DCF-85DE-C7ED0EE1CAAD' 1 row(s) affected 0.047 sec
```

The output pane shows a single row of data indicating the execution of the delete query and its success.

## 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.

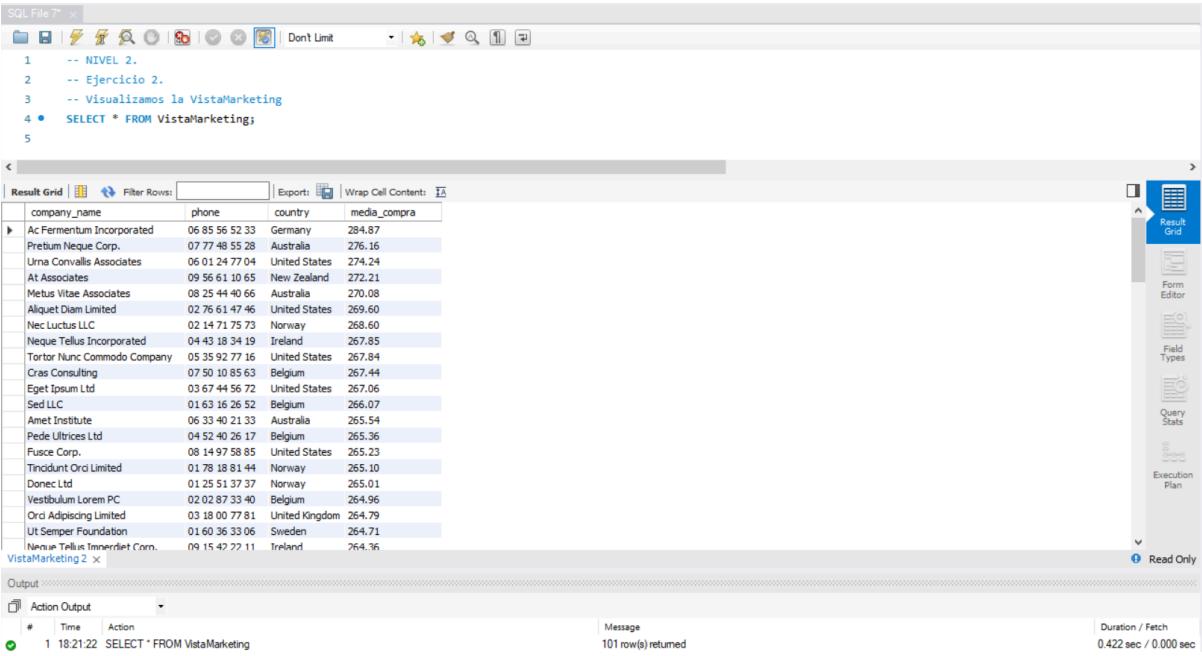


```
-- NIVEL 2.
-- Ejercicio 2.
-- Creamos la VistaMarketing
CREATE VIEW VistaMarketing AS
SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount),2) AS media_compra
FROM company c
JOIN transaction t ON c.id=t.company_id
GROUP BY c.company_name, c.phone, c.country
ORDER BY media_compra DESC;
```

The screenshot shows the SQL Server Management Studio interface. The code above is entered into the query window. Below the code, the output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	18:20:08	CREATE VIEW VistaMarketing AS SELECT c.company_name, c.phone, c.country, ROUND(AVG(t.amount),2)... 0 row(s) affected		0.046 sec

Para visualizar la información de la vista utilizamos un SELECT:



```
-- NIVEL 2.
-- Ejercicio 2.
-- Visualizamos la VistaMarketing
SELECT * FROM VistaMarketing;
```

The screenshot shows the SQL Server Management Studio interface. The code above is entered into the query window. Below the code, the result grid displays the data from the VistaMarketing view:

company_name	phone	country	media_compra
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
Aliquet Diam Limited	02 76 61 47 46	United States	269.60
Nec Luctus LLC	02 14 71 75 73	Norway	268.60
Neque Tellus Incorporated	04 43 18 34 19	Ireland	267.85
Tortor Nunc Commodo Company	05 35 92 77 16	United States	267.84
Cras Consulting	07 50 10 85 63	Belgium	267.44
Eget Ipsum Ltd	03 67 44 56 72	United States	267.06
Sed LLC	01 63 16 26 52	Belgium	266.07
Amet Institute	06 33 40 21 33	Australia	265.54
Pede Ultrices Ltd	04 52 40 26 17	Belgium	265.36
Fusce Corp.	08 14 97 58 85	United States	265.23
Tincidunt Ord Limited	01 78 18 81 44	Norway	265.10
Donec Ltd	01 25 51 37 37	Norway	265.01
Vestibulum Lorem PC	02 02 87 33 40	Belgium	264.96
Orch Adipiscing Limited	03 18 00 77 81	United Kingdom	264.79
Ut Semper Foundation	01 60 36 33 06	Sweden	264.71
Naveue Tellus Immerdiet Conn.	09 15 47 22 11	Ireland	264.36

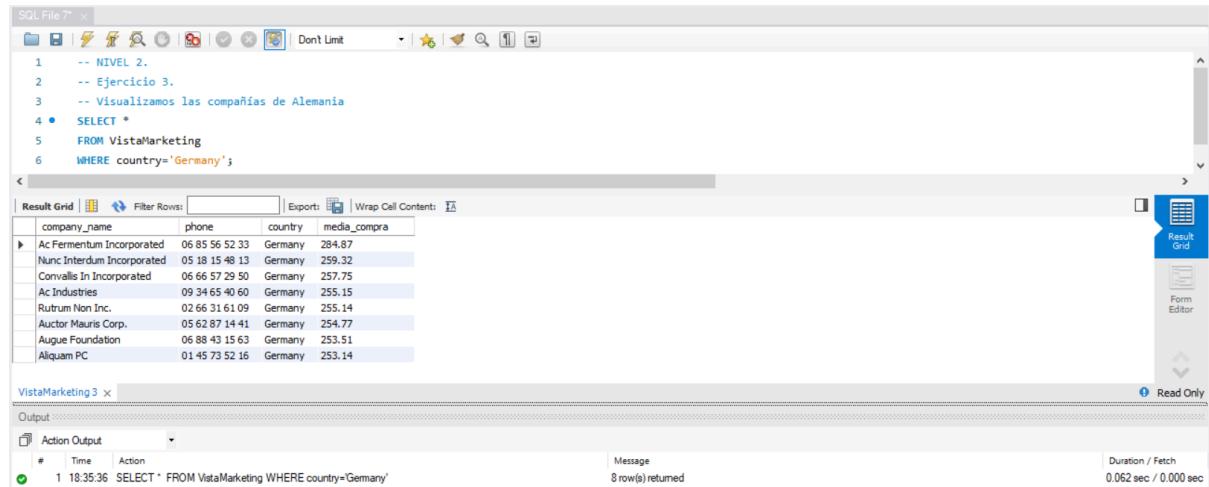
The output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	18:21:22	SELECT * FROM VistaMarketing	101 row(s) returned	0.422 sec / 0.000 sec

## 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.



The screenshot shows a SQL IDE interface with the following details:

- SQL File 7\***: The current file is named "7".
- Code Area (SQL):**

```
1 -- NIVEL 2.
2 -- Ejercicio 3.
3 -- Visualizamos las compañías de Alemania
4 • SELECT *
5 FROM VistaMarketing
6 WHERE country='Germany';
```
- Result Grid:** A table displaying company data where the country is Germany. The columns are company\_name, phone, country, and media\_compra. The data includes:

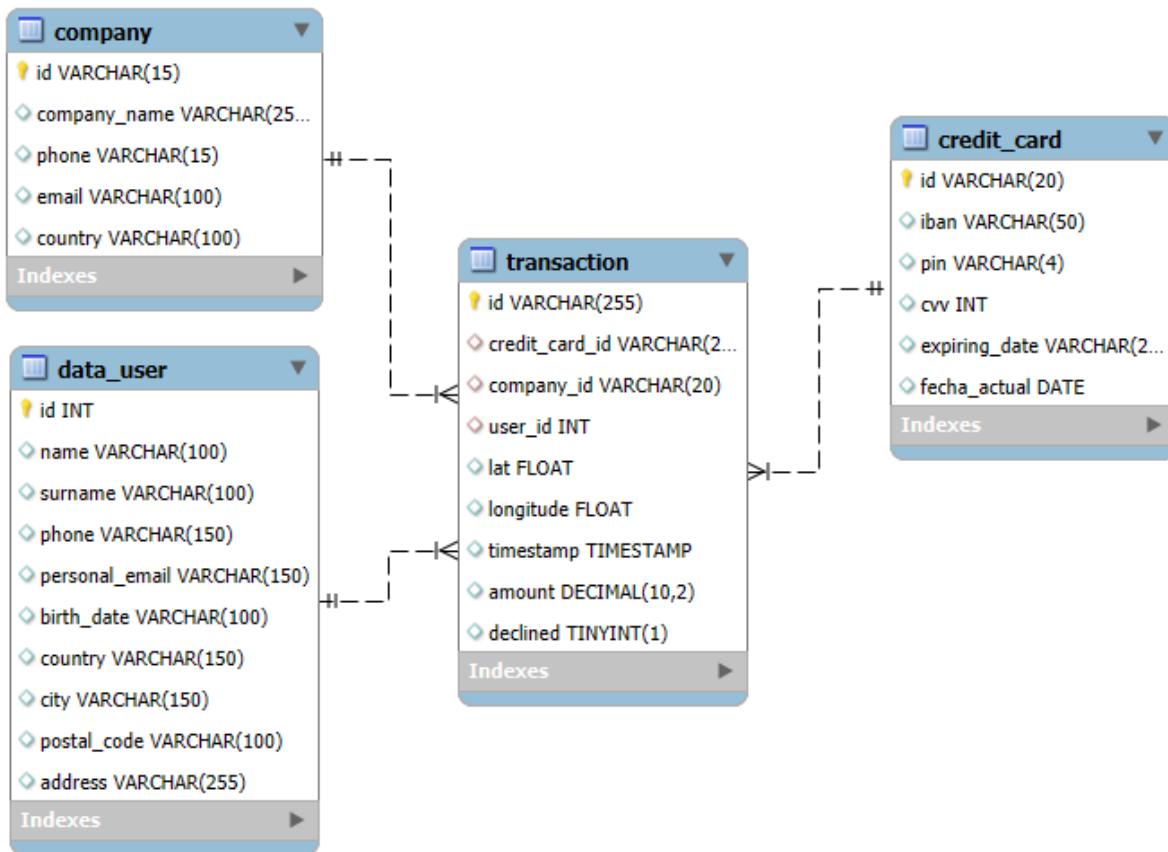
company_name	phone	country	media_compra
Ac Fermentum Incorporated	06 85 56 52 33	Germany	284.87
Nunc Interdum Incorporated	05 18 15 48 13	Germany	259.32
Convalis In Incorporated	06 66 57 29 50	Germany	257.75
Ac Industries	09 34 65 40 60	Germany	255.15
Rutrum Non Inc.	02 66 31 61 09	Germany	255.14
Auctor Mauris Corp.	05 62 87 14 41	Germany	254.77
Augue Foundation	06 88 43 15 63	Germany	253.51
Aliquam PC	01 45 73 52 16	Germany	253.14
- Output Panel:** Shows the executed query and its results.

#	Time	Action	Message	Duration / Fetch
1	18:35:36	SELECT * FROM VistaMarketing WHERE country='Germany'	8 row(s) returned	0.062 sec / 0.000 sec

# 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.

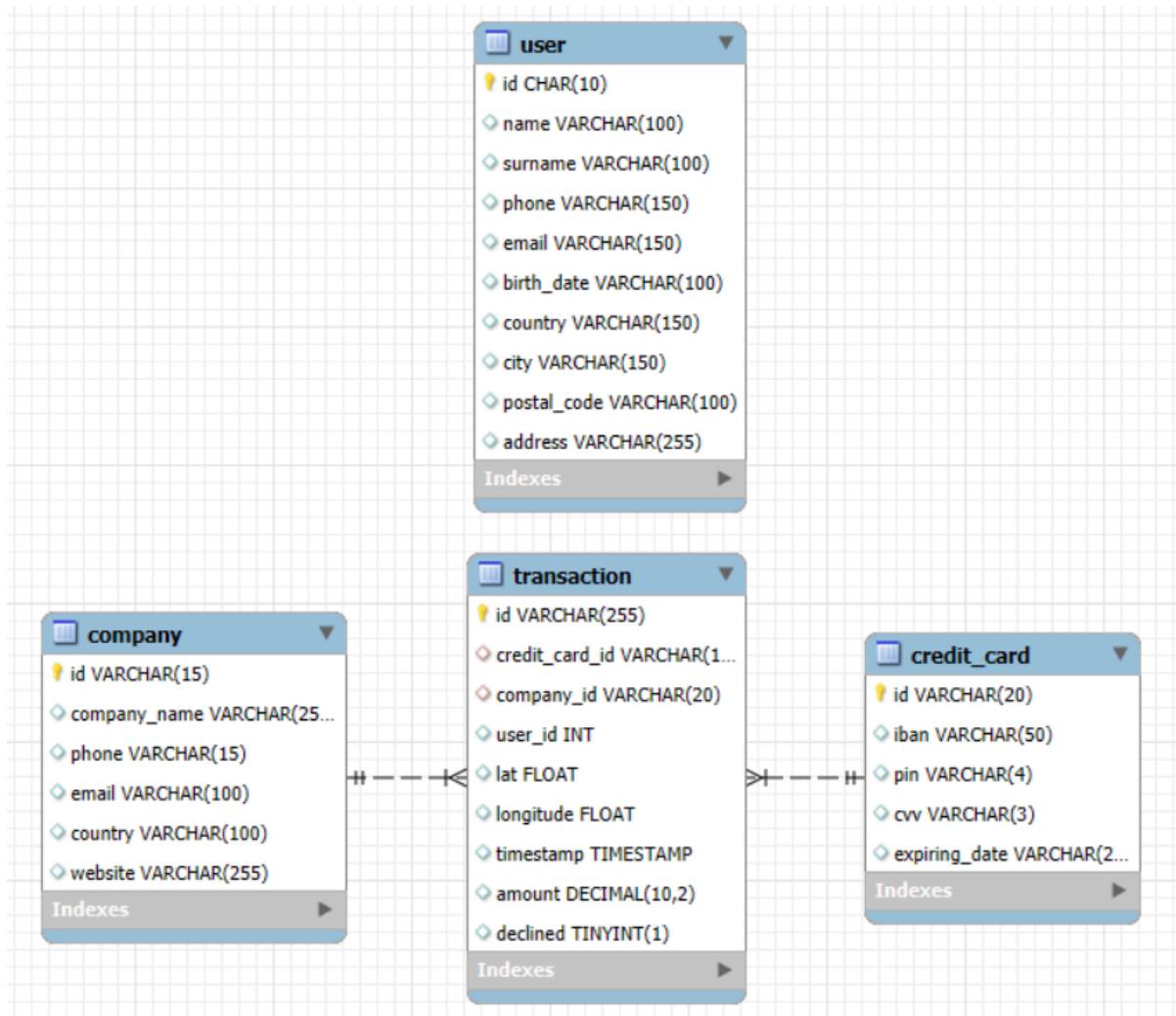
Paso 1. Creamos la tabla user a partir del archivo de la estructura:

The screenshot shows the MySQL Workbench interface with a SQL editor window titled "SQL File 7" containing the following SQL code:

```
1 -- NIVEL 3,
2 -- Ejercicio 1.
3 -- Paso 1. Creamos la tabla user
4 CREATE TABLE IF NOT EXISTS user (
5     id CHAR(10) PRIMARY KEY,
6     name VARCHAR(100),
7     surname VARCHAR(100),
8     phone VARCHAR(150),
9     email VARCHAR(150),
10    birth_date VARCHAR(100),
11    country VARCHAR(150),
12    city VARCHAR(150),
13    postal_code VARCHAR(100),
14    address VARCHAR(255)
15 );
```

The output pane shows the command was executed at 21:29:32 and affected 0 rows.

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

SQL File 7 - datos introducir sprint3 user.x

Dont Limit

```
1 -- Paso 2. Insertamos los datos
2 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
3 •     151, "Meghan", "Hayden", "0800 746 6747", "arcu.vel@h
4 •     152, "Hakeem", "Alford", "(0111) 367 0184", "adipisci
5 •     153, "Keegan", "Pugh", "(016977) 3851", "sodales.nisi
6 •     154, "Cooper", "Bullock", "(021) 2521 6627", "et@outl
7 •     155, "Joshua", "Russell", "065 4409 5286", "justo.nec
8 •     156, "Remedios", "Case", "055 3114 1566", "mollis.pha
9 •     157, "Philip", "Carey", "0800 640 6251", "phasellus@y
10 •    158, "Fatima", "Dyer", "0800 1111", "adipiscing@google
11 •    159, "Kyllynn", "Acedovo", "056 5727 9602", "dignissim
12 •    160, "Lael", "Moody", "07123 850737", "nunc.sed@aol.o
13 •    161, "Nora", "Reeves", "(01692) 29410", "ac@aol.com",
14 •    162, "Francesca", "Sawyer", "(01632) 322797, "nunc@pr
15 •    163, "Denton", "Blackburn", "(015406) 66385", "tincid
16 •    164, "Preston", "Hood", "0845 46 47", "convallis.est.
165, "Nora", "Cantrell", "0500 767716", "varius.et.eu
```

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:

```
SQL File 7 > 
File Edit View Insert Tools Window Help
1 -- NIVEL 3.
2 -- Ejercicio 1.
3 -- Paso 3. Tabla "user"
4 -- 3.1. Cambiar el nombre a data_user
5 • ALTER TABLE user
6 RENAME TO data_user;

Output
Action Output
# Time Action Message Duration / F
1 21:49:51 ALTER TABLE user RENAME TO data_user 0 row(s) affected 0.079 sec
```

### 3.2. Cambiar tipo de “id”:

SQL File 7

```
1 -- NIVEL 3.  
2 -- Ejercicio 1.  
3 -- Paso 3. Tabla "user"  
4 -- 3.2. Cambiar tipo de "id"  
5 • ALTER TABLE data_user  
6 MODIFY COLUMN id INT;
```

Output

Action Output
# Time Action
1 21:55:36 ALTER TABLE data_user MODIFY COLUMN id INT

Message
5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

Duration / Fetch
0.687 sec

### 3.3. Cambiar el nombre de “email”

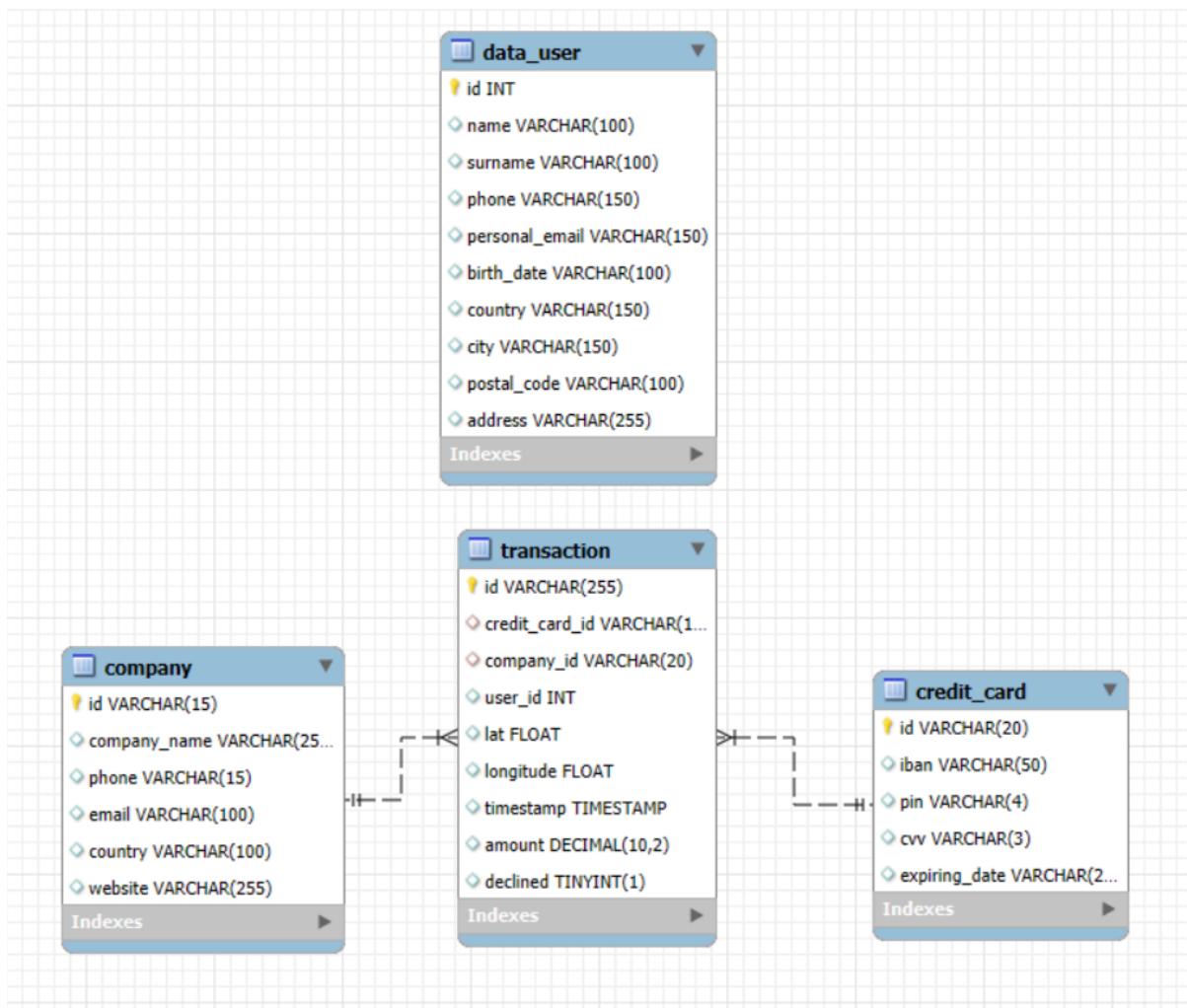
The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following SQL code is displayed:

```
1 -- NIVEL 3.
2 -- Ejercicio 1.
3 -- Paso 3. Tabla "user"
4 -- 3.3. Cambiar el nombre de "email" a "personal_email"
5 • ALTER TABLE data_user
6 RENAME COLUMN email to personal_email;
```

In the Output tab, the results of the execution are shown:

Action	Time	Message	Duration / Fetch
1 21:59:23		ALTER TABLE data_user RENAME COLUMN email to personal_email 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.110 sec

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 caracteres de “credit\_card\_id”.

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

```
-- NIVEL 3.  
-- Ejercicio 1.  
-- Paso 4. Tabla "transaction"  
-- 4.1. Creamos la conexión con "data_user"  
ALTER TABLE transaction  
ADD CONSTRAINT fk_transaction_data_user  
FOREIGN KEY (user_id) REFERENCES data_user(id);
```

Output

#	Time	Action	Message	Duration / Fetch
1	22:09:49	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_data_user FOREIGN KEY (user_id) REFEREN...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('transactions'.#sql-1acc_57...)	1.188 sec

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

```
SELECT DISTINCT user_id  
FROM transaction  
WHERE user_id NOT IN (SELECT id FROM data_user);
```

Result Grid

user_id
9999

Output

#	Time	Action	Message	Duration / Fetch
1	22:13:42	SELECT DISTINCT user_id FROM transaction WHERE user_id NOT IN (SELECT id FROM data_user)	1 row(s) returned	0.297 sec / 0.000 sec

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

```
-- Insertamos el usuario a la tabla data_user  
INSERT INTO data_user (id)  
VALUES (9999);
```

Output

#	Time	Action	Message	Duration / Fetch
1	22:25:41	INSERT INTO data_user (id) VALUES (9999)	1 row(s) affected	0.000 sec

Creamos la conexión con “transaction”:

```
-- Creamos la conexión con "data_user"  
ALTER TABLE transaction  
ADD CONSTRAINT fk_transaction_data_user  
FOREIGN KEY (user_id) REFERENCES data_user(id);
```

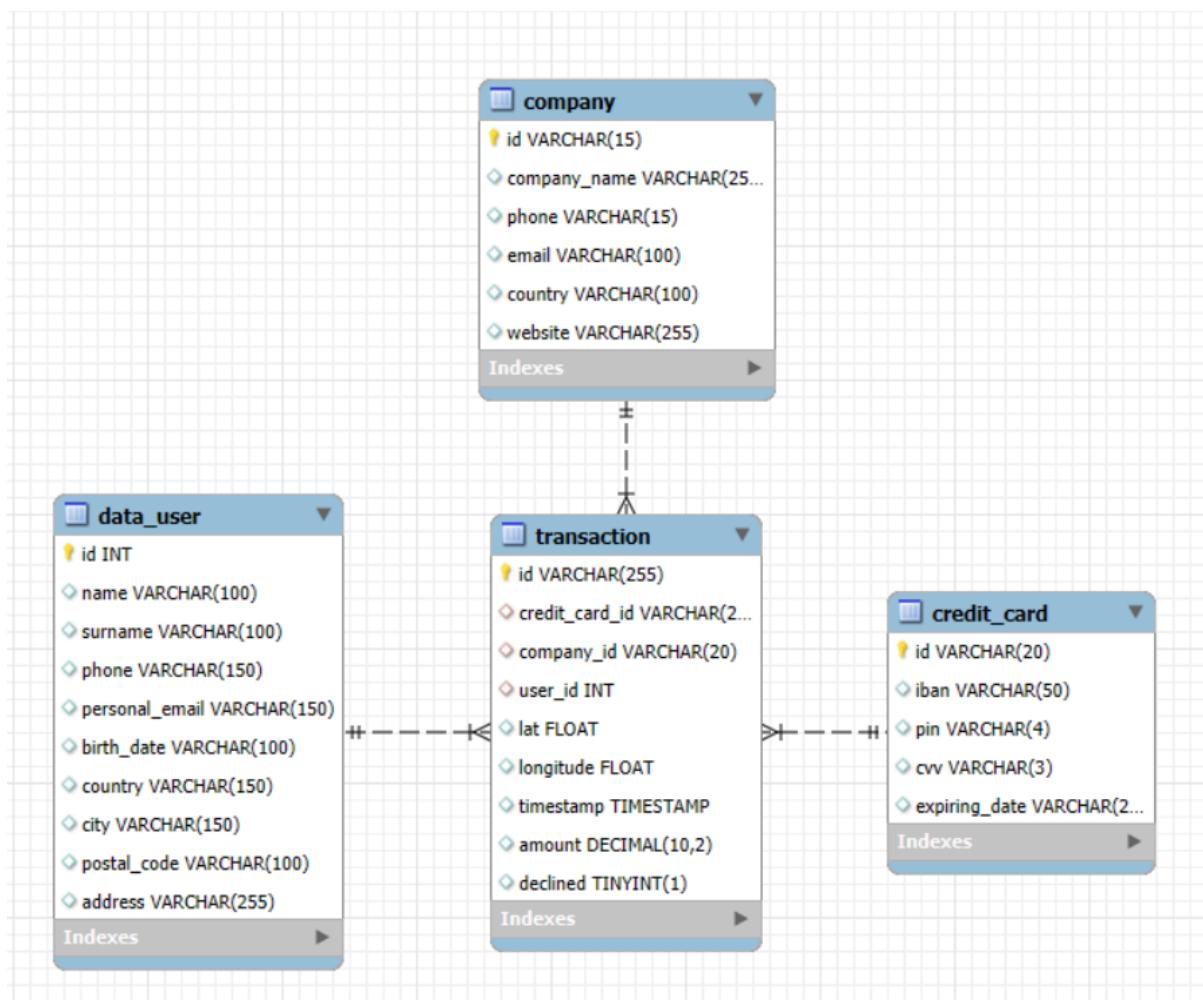
Output

#	Time	Action	Message	Duration / Fetch
1	22:26:45	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_data_user FOREIGN KEY (user_id) REFEREN...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0	7.531 sec

Cambiamos la longitud de “credit\_card\_id”:

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a single query:1 -- Cambiamos la longitud de "credit\_card\_id";
2 • ALTER TABLE transaction
3 MODIFY credit\_card\_id VARCHAR(20);
4In the Output tab, the results of the query are displayed:1 22:29:29 ALTER TABLE transaction MODIFY credit\_card\_id VARCHAR(20)Details about the execution are shown at the bottom:Message: 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
Duration / Fetch: 0.125 sec

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:

```

SQL File ? ×
File | Edit | View | Tools | Help | Don't Limit | 
1 -- NIVEL 3.
2 -- Ejercicio 1.
3 -- Paso 5. Tabla "credit_card"
4 -- 5.1. Cambiar CVV de VARCHAR a INT
5 • ALTER TABLE credit_card
6   MODIFY cvv INT;
7

```

**Output**

Action Output	#	Time	Action	Message	Duration / Fetch
1 22:35:39	ALTER TABLE credit_card MODIFY cvv INT			5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0	0.516 sec

Añadir la columna “fecha\_actual”:

```

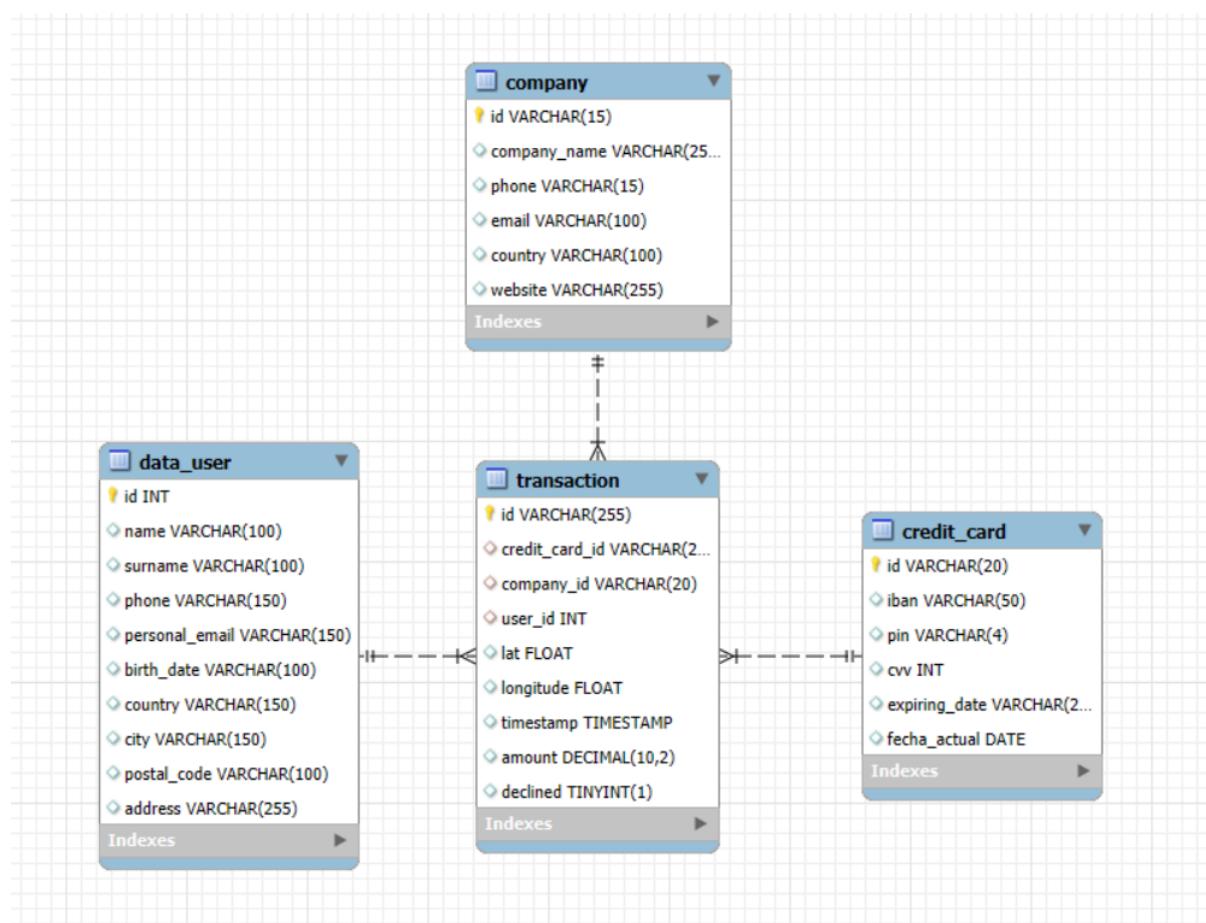
SQL File ? ×
File | Edit | View | Tools | Help | Don't Limit | 
1 -- NIVEL 3.
2 -- Ejercicio 1.
3 -- Paso 5. Tabla "credit_card"
4 -- 5.2. Añadir la columna "fecha_actual" de tipo DATE
5 • ALTER TABLE credit_card
6   ADD fecha_actual DATE;
7

```

**Output**

Action Output	#	Time	Action	Message	Duration / Fetch
1 22:38:15	ALTER TABLE credit_card ADD fecha_actual DATE			0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec

Observamos en el esquema los cambios realizados en la tabla “credit\_card”:



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

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, the following SQL code is written:

```
1 -- NIVEL 3.
2 -- Ejercicio 1.
3 -- Paso 6. Tabla "company"
4 -- 6.1. Eliminar la columna "website"
5 • ALTER TABLE company
6 DROP COLUMN website;
```

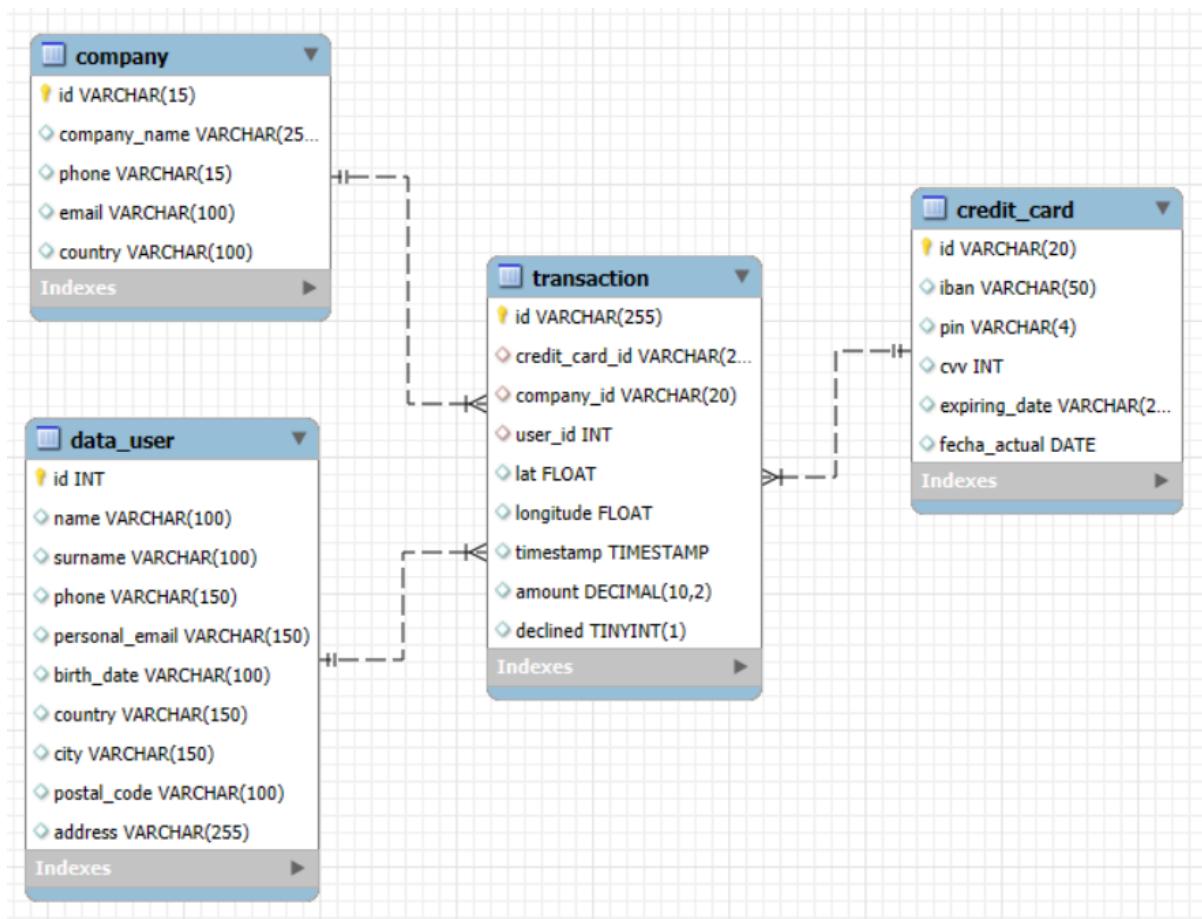
In the Output tab, the results of the execution are shown:

#	Time	Action
1	22:43:49	ALTER TABLE company DROP COLUMN website

Message: 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Duration / Fetch: 0.156 sec

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



## Ejercicio 2

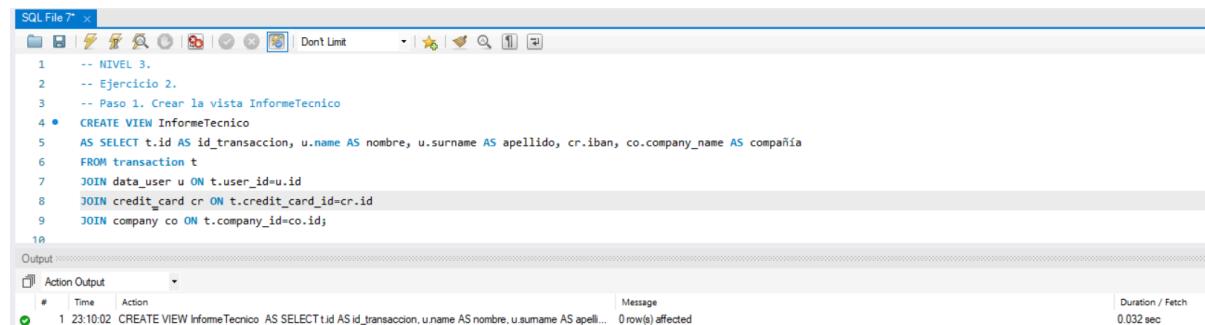
La empresa también le pide crear una vista llamada "InformeTecnico" que contenga la siguiente información:

- 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.



The screenshot shows the SQL Developer interface with a SQL File tab open. The code in the editor is as follows:

```
-- NIVEL 3.  
-- Ejercicio 2.  
-- Paso 1. Crear la vista InformeTecnico  
CREATE VIEW InformeTecnico  
AS SELECT t.id AS id_transaccion, u.name AS nombre, u.surname AS apellido, cr.iban, co.company_name AS compañia  
FROM transaction t  
JOIN data_user u ON t.user_id=u.id  
JOIN credit_card cr ON t.credit_card_id=cr.id  
JOIN company co ON t.company_id=co.id;
```

The output pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	23:10:02	CREATE VIEW InformeTecnico AS SELECT t.id AS id_transaccion, u.name AS nombre, u.surname AS apellido, cr.iban, co.company_name AS compañia FROM transaction t JOIN data_user u ON t.user_id=u.id JOIN credit_card cr ON t.credit_card_id=cr.id JOIN company co ON t.company_id=co.id;	0 row(s) affected	0.032 sec

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

The screenshot shows the MySQL Workbench interface. At the top, there's a toolbar with various icons like file, edit, search, and database selection. Below the toolbar is a status bar with 'Dont Limit' and other information. The main area has tabs for 'Result Grid' and 'Form Editor'. The 'Result Grid' tab is active, displaying a table with columns: id\_transaccion, nombre, apellido, iban, and compañia. The table contains approximately 100 rows of data. To the right of the grid, there are several small windows for 'Field Types', 'Query Stats', and 'Execution Plan'. At the bottom, there's an 'Output' section with a table showing the execution of the query: 'Action Output' (1 row selected), 'Message' (100000 row(s) returned), and 'Duration / Fetch' (2.250 sec / 0.156 sec). A note at the bottom says 'Read Only'.

```
SQL File 7* ×
1 -- NIVEL 3.
2 -- Ejercicio 2.
3 -- Paso 2. Mostrar la vista ordenando por id
4 • SELECT *
5 FROM InformeTecnico
6 ORDER BY id_transaccion DESC;
7
```

	id_transaccion	nombre	apellido	iban	compañia
▶	FFFF31D6-9495-47CE-B54A-7D88E1CC274B	Bmrgli	Tprvrmrc	XX794814451211289182490922	Turpe Company
	FFFFCF76D-EFCF0-4985-A2D0-B2A7B75998FC	Dfired	Vlqjqlj	XX636251701647892036676034	Amet Nulla Donec Corporation
	FFFC9E80-27C7-4AD6-98F2-7533EF4DF126	Securi	Faofvqfy	XX162677143304223631437567	Nunc Interdum Incorporated
	FFFFB2700-F53A-4D5D-9666-E5307C53C848	Gggpa	Urzjulh	XX39511426708201995267052	Viverra Donec Foundation
	FFF9E3CE-234E-408C-A8EF-F9CA05772244	Yshimq	Zpsjeed	XX045462156537570367941	Convallis In Incorporated
	FFF9E178-6CD2-4D93-9980-49AE06808981	Jevepx	Xwcviznmh	XX321405515711654384711481	Mus Aenean Eget Foundation
	FFF867C9-17B5-4B1F-APD9-F8023AA449E	Fqjgnd	Lvhfxyl	XX278446342932680979729426	Cras Vehicula Aliquet Industries
	FFF7042D-18C5-4D0D-832C-4D90A4AC8F26	Njoraa	Egspcuil	XX405009272572550082027209	Placerat LLP
	FFF660D-4244-47F6-9210-E5D1DCB990B0	Lopzaj	Itgrfay	XX6376659736627454015125	Pede Cum Ltd
	FFF5C660-4441-436D-B027-E6C53B618622	Gmnbu	Oxdvhkl	XX237620256172646394016483	At Associates
	FFF54F54-B439-41F0-BD02-F7332DC1ACAB	Gqcffy	Mpfrlbn	XX802723943240147612158718	Erin Condimentum Ltd
	FFF42F7D-7A0D-4E62-AF72-5996904FBAA9	Ddklugu	Ysbpry	XX926442301555195974199541	At Pede Corp.
	FFF42620-1968-4A1F-B9D7-27843CB097A5	Sbjrvp	Eignzoff	XX395457232638959102336873	Amet Nulla Donec Corporation
	FFF290EB-79D2-497C-BDCE-2EC63FEE3E9AD	Wcfrys	Dlkthgtjd	XX458989048222230717519935	Integer Mollis Corp.
	FFF20FB4-9014-4AC3-B938-E31E7786C49A	Afbmt	Yshvmtk	XX148483552743443735712134	Ac Fermentum Incorporated
	FFF205DF-60FD-4E49-B337-EE476513A96C	Gcbnpr	Rsbbojqg	XX401515215969396565954528	Pretium Neque Corp.
	FFF11165-302A-46F6-B5D1-11A80FEBED65	Sxmbuj	Fofmhlyy	XX984089430672237985516070	Augue Foundation
	FFF5F9FA-D931-41A2-R45F-789X9F9K4FFR	Frknif	Durhfrxt	XX371816115649671774181111	Orak Vehicula Aliquet Industries

InformeTecnico 4 ×

Output

Action	Output
#	Action Output
1	Time 23:13:25 Action SELECT * FROM InformeTecnico ORDER BY id_transaccion DESC

Message 100000 row(s) returned Duration / Fetch 2.250 sec / 0.156 sec

Read Only