Sprint 3: Manipulación de tablas

Santiago Álvarez Salvado

Contenido

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.

Para crear la tabla credit_card se ha utilizado un ejemplo del archivo adjunto datos_introducir_sprint3_credit.sql, el tipo de dato y la longitud será varchar(255) para todo porque aún no se sabe la longitud exacta, posteriormente se introducen los datos del archivo datos_introducir_sprint3_credit.sql y se define la foreign key para unir las tablas con credit_Card_id de transaction e id de credit_card.

```sql
create table if not exists credit_card (
    id varchar(255) primary key not null,
    iban varchar(255),
    pan varchar(255),
    pin varchar(255),
    cvv varchar(255),
    expiring_date varchar(255)
);
```

Output

Action Output

	#	Time	Action	Message
✓	1	21:49:16	create table if not exists credit_card (id varchar(255) primary key not null, iban varchar(255), pan varchar(255)...	0 row(s) affected

Se ejecuta el archivo datos_introducir_sprint3_credit.sql para cargar los datos en la tabla credit_card;

```sql
-- Insertamos datos de credit_card
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984', '10/30/22');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2945', 'DO26854763748537475216568689', '5142423821948828', '9080', '887', '08/24/23');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2952', 'BG45IVQL52710525608255', '4556 453 55 5287', '4598', '438', '06/29/21');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2959', 'CR7242477244335841535', '372461377349375', '3583', '667', '02/24/23');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2966', 'BG72LKTQ15627628377363', '448566 886747 7265', '4900', '130', '10/29/24');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2973', 'PT87806228135092429456346', '544 58654 54343 384', '8760', '887', '01/30/25');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2980', 'DE39241881883086277136', '402400 7145845969', '5075', '596', '07/24/22');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2987', 'GE89681434837748781813', '3763 747687 76666', '2298', '797', '10/31/23');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2994', 'BH62714428368066765294', '344283273252593', '7545', '595', '02/28/22');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3001', 'CY49087426654774581266832110', '511722 924833 2244', '9562', '867', '09/16/22');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3008', 'LU507216693616119230', '4485744464433884', '1856', '740', '04/05/25');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3015', 'PS119398216295715968342456821', '3784 662233 17389', '3246', '822', '01/31/22');
INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3022', 'GT91695162850556977423121857', '5164 1379 4842 3951', '5610', '342', '04/25/25');
```

Output

Action Output

	#	Time	Action	Message	Duration / Fetch
✓	4991	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9571', 'XX48491538243714199...	1 row(s) affected	0.000 sec
✓	4992	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9572', 'XX40544818626973038...	1 row(s) affected	0.000 sec
✓	4993	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9573', 'XX96533893105308890...	1 row(s) affected	0.000 sec
✓	4994	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9574', 'XX66276175836143268...	1 row(s) affected	0.000 sec
✓	4995	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9575', 'XX49988121607362657...	1 row(s) affected	0.000 sec
✓	4996	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9576', 'XX52971093058771264...	1 row(s) affected	0.000 sec
✓	4997	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9577', 'XX15891407859480863...	1 row(s) affected	0.000 sec
✓	4998	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9578', 'XX99153964645611056...	1 row(s) affected	0.000 sec
✓	4999	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9579', 'XX29639309158717020...	1 row(s) affected	0.000 sec
✓	5000	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9580', 'XX78125888985195080...	1 row(s) affected	0.000 sec
✓	5001	21:54:27	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcS-9581', 'XX91567051640538812...	1 row(s) affected	0.000 sec

Se verifica la longitud de todas las columnas con max(length())

```
18 • select max(length(id)) as longitud_id from credit_card;
19
```

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

longitud_id
8

Result 2

Output

Action Output

#	Time	Action	Message
1	22:00:07	select max(length(id)) as longitud_id from credit_card	1 row(s) returned

```
20 • select max(length(iban)) as longitud_iban from credit_card;
21
```

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

longitud_iban
31

Result 3

Output

Action Output

#	Time	Action	Message
1	22:00:07	select max(length(id)) as longitud_id from credit_card	1 row(s) returned
2	22:00:30	select max(length(iban)) as longitud_iban from credit_card	1 row(s) returned

```
22 • select max(length(pan)) as longitud_pan from credit_card;
23
```

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

longitud_pan
19

Result 4

Output

Action Output

#	Time	Action	Message
1	22:00:07	select max(length(id)) as longitud_id from credit_card	1 row(s) returned
2	22:00:30	select max(length(iban)) as longitud_iban from credit_card	1 row(s) returned
3	22:00:52	select max(length(pan)) as longitud_pan from credit_card	1 row(s) returned

```
24 •  select max(length(pin)) as longitud_pin from credit_card;
25
```

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

longitud_pin
4

Result 5

Output

Action Output

#	Time	Action	Message
2	22:00:30	select max(length(iban)) as longitud_iban from credit_card	1 row(s) returned
3	22:00:52	select max(length(pan)) as longitud_pan from credit_card	1 row(s) returned
4	22:01:20	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned

```
26 •  select max(length(cvv)) as longitud_cvv from credit_card;
27
```

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

longitud_cvv
3

Result 6

Output

Action Output

#	Time	Action	Message
3	22:00:52	select max(length(pan)) as longitud_pan from credit_card	1 row(s) returned
4	22:01:20	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned
5	22:02:23	select max(length(cvv)) as longitud_cvv from credit_card	1 row(s) returned

```
28 •  select max(length(expiring_date)) as longitud_fecha_expiracion from credit_card;
29
```

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

longitud_fecha_expiracion
8

Result 7

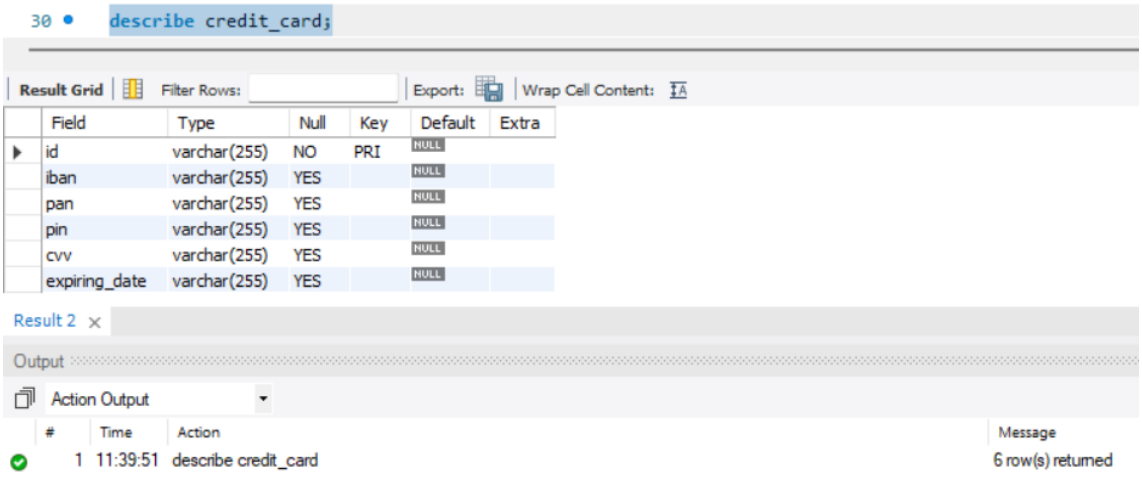
Output

Action Output

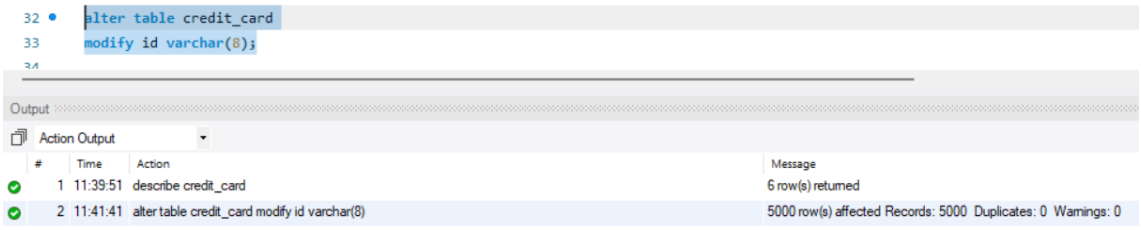
#	Time	Action	Message
4	22:01:20	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned
5	22:02:23	select max(length(cvv)) as longitud_cvv from credit_card	1 row(s) returned
6	22:02:58	select max(length(expiring_date)) as longitud_fecha_expiracion from credit_card	1 row(s) returned

En un entorno real esto no tendría mucho sentido porque los datos actuales no garantizan los futuros, es decir, que si se le da una longitud que marca cada campo, y entra un dato más grande que lo establecido, dará error y causará problemas, lo correcto sería establecer una longitud adecuada, sin exceder ni quedarse corto, salvo los campos que sean datos cerrados como pin, pan, cvv, iban, dni, etc; por lo que en base a la longitud que se ha obtenido se pondrá ajustada para este ejercicio, en un caso real quizá sería mejor poner el doble de la longitud obtenida por si entrase un dato más grande que la longitud máxima obtenida.

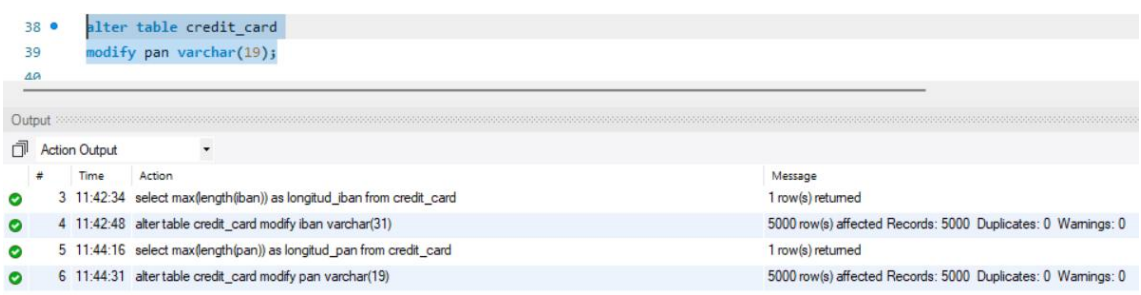
Antes de modificar la longitud se puede observar la tabla con el comando describe y ver la longitud actual



Ahora se modifica la longitud y el tipo de dato según sea necesario



En un caso real el iban no se guardaría completo, se guardarían los 4 últimos dígitos.



El pin tampoco se guardaria por seguridad, y si se guarda tendria que estar encriptado o protegido de alguna manera.

```
41 •  alter table credit_card
42    modify pin varchar(4);
43
```

Output

Action Output

#	Time	Action	Message
5	11:44:16	select max(length(pan)) as longitud_pan from credit_card	1 row(s) returned
6	11:44:31	alter table credit_card modify pan varchar(19)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
7	11:44:56	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned
8	11:45:08	alter table credit_card modify pin varchar(4)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

El cvv ocurriria lo mismo que con el pin.

```
44 •  alter table credit_card
45    modify cvv varchar(3);
46
```

Output

Action Output

#	Time	Action	Message
7	11:44:56	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned
8	11:45:08	alter table credit_card modify pin varchar(4)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
9	11:45:38	select max(length(cvv)) as longitud_cvv from credit_card	1 row(s) returned
10	11:45:47	alter table credit_card modify cvv varchar(3)	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

El campo expiring_date será necesario convertirlo a date, al observar los datos se puede ver que la fecha está en formato MM/DD/YY, el formato de date es DD/MM/YYYY o MM/DD/YYYY, al tener dos dígitos la fecha, si los datos son muy antiguos podría causar ambigüedad, ya que sql no sabe diferenciar si por ejemplo 15 seria 1915 o 2015, normalmente se suele guardar con el último día del mes, ya que las tarjetas de crédito no caducan un día concreto sino en un mes concreto, normalmente al final del mes que marque la tarjeta, para realizar esto sería necesario actualizar el tipo de dato a date, poner el ultimo día con last_day y str_to_date para la conversión del dato de varchar a date, limit 6000 se pone porque si no salta el modo seguro de mysql a la hora de realizar update, aunque con 5000 (que son los registros que tiene la tabla) sería suficiente.

```
50 •  UPDATE credit_card
51    SET credit_card.expiring_date =
52        LAST_DAY(
53            STR_TO_DATE(credit_card.expiring_date, '%m/%d/%y')
54        )
55    limit 6000;
56
```

Output

Action Output

#	Time	Action	Message
1	13:13:10	UPDATE credit_card SET credit_card.expiring_date = LAST_DAY(STR_TO_DATE(credit_card.expiring...	5000 row(s) affected Rows matched: 5000 Changed: 5000 Warnings: 0

Por último se modifica el dato a tipo date porque sigue como varchar(255).

```
54 • alter table credit_card
55   modify expiring_date date;
56
```

Output

Action Output

#	Time	Action	Message
1	13:17:57	alter table credit_card modify expiring_date date	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

Se vuelve a comprobar los datos de la tabla para ver que se han modificado los campos correctamente.

```
57 • describe credit_card;
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(8)	NO	PRI	NULL	
iban	varchar(31)	YES		NULL	
pan	varchar(19)	YES		NULL	
pin	varchar(4)	YES		NULL	
cvv	varchar(3)	YES		NULL	
expiring_date	date	YES		NULL	

Result 4

Output

Action Output

#	Time	Action	Message
1	13:17:57	alter table credit_card modify expiring_date date	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
2	13:21:07	describe credit_card	6 row(s) returned

Se declara la clave foránea de la tabla credit_card en la tabla transaction

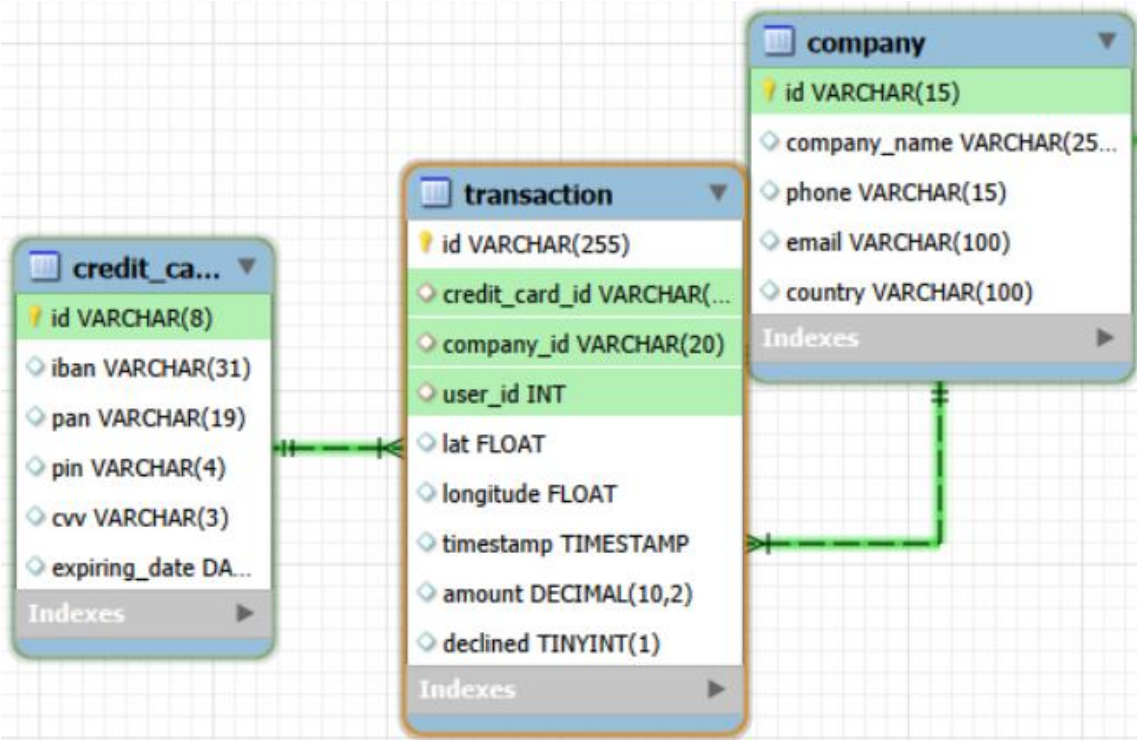
```
59 • alter table transaction
60   add constraint fk_credit_cardid
61   foreign key (credit_card_id)
62   references credit_card(id);
63
```

Output

Action Output

#	Time	Action	Message
1	16:35:10	alter table transaction add constraint fk_credit_cardid foreign key (credit_card_id) references credit_card(id)	99999 row(s) affected Records: 99999 Duplicates: 0 Warnings: 0

Para generar el diagrama entidad relación se ha utilizado la herramienta reverse engineer de workbench; como se puede observar, aparecen las tablas anteriores (transaction y company), y la nueva tabla credit_card, la cual se relaciona con transaction a través del campo credit_card_id de la tabla transaction y el campo id de la tabla credit card, la relación entre ambas tablas es 1-N desde credit_card a transacction, ya que una tarjeta de crédito puede tener muchas transacciones, pero una transacción solo puede pertenecer a una tarjeta de créditos.



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.

Para este apartado primero se verifican los datos para asegurar que realmente hay un problema con el número de cuenta asociado a esa tarjeta, para ello se realiza una selección de toda la información de la tabla credit_card y se filtra con el id que está causando el posible problema, como se puede observar, efectivamente el número de cuenta no coincide.

```
71 •  select * from credit_card
72    where id = "CcU-2938";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR301950312213576817638661	5424465566813633	3257	984	2022-10-31
NULL	NULL	NULL	NULL	NULL	NULL

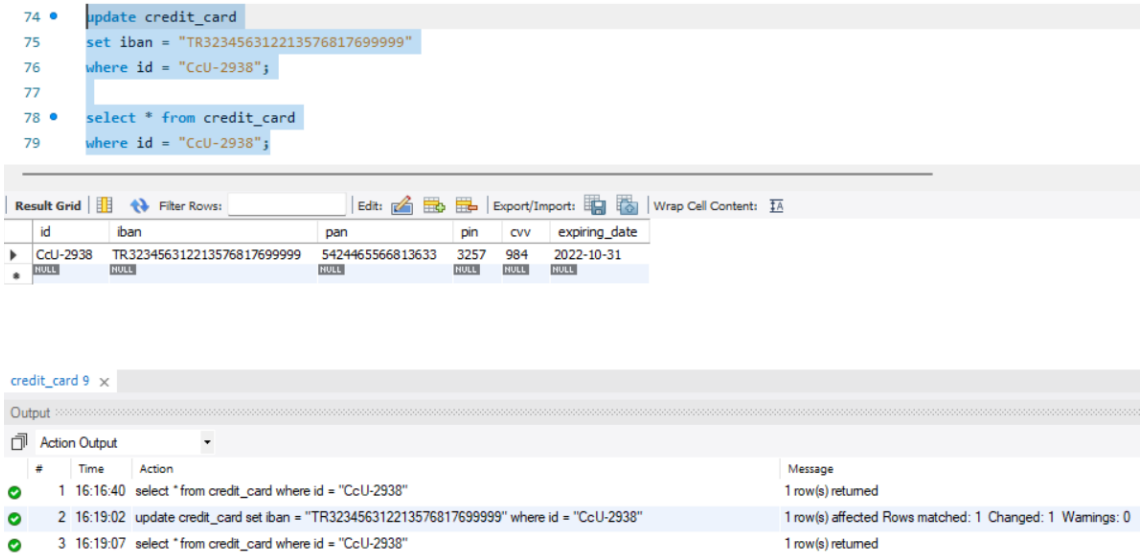
credit_card 8

Output

Action Output

#	Time	Action	Message
1	16:16:40	select * from credit_card where id = "CcU-2938"	1 row(s) returned

Para poder solucionar este problema se actualiza la tabla con update y el nombre de la tabla, y se declara el nuevo valor, para que surta efecto habrá que poner un filtro en el que se especifique a que id se le hará el cambio de número de cuenta, posteriormente al ejecutar la query anterior se podrá observar que ahora el número de cuenta coincide con el que identifico el departamento de recursos humanos.



```
74 •  update credit_card
75    set iban = "TR3234563122135768176999999"
76    where id = "CcU-2938";
77
78 •  select * from credit_card
79    where id = "CcU-2938";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	iban	pan	pin	cvv	expiring_date
CcU-2938	TR3234563122135768176999999	5424465566813633	3257	984	2022-10-31
NULL	NULL	NULL	NULL	NULL	NULL

credit_card 9

Output

Action Output

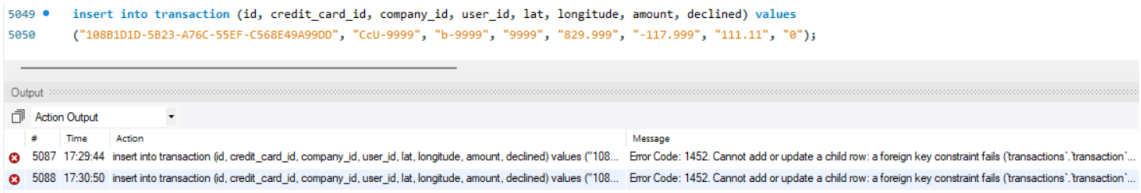
#	Time	Action	Message
1	16:16:40	select * from credit_card where id = "CcU-2938"	1 row(s) returned
2	16:19:02	update credit_card set iban = "TR3234563122135768176999999" where id = "CcU-2938"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
3	16:19:07	select * from credit_card where id = "CcU-2938"	1 row(s) returned

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
lato	829.999
longitud	-117.999
amunt	111.11
declined	0

HAY ALGUNOS CAMPOS MAL ESCRITOS EN LA TABLA (lato , longitud y amunt, serían lat, longitude y amount) Y FALTA timestamp, además, al hacer esto ocurriría un error como el siguiente: Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`transactions`.`transaction`, CONSTRAINT `transaction_ibfk_1` FOREIGN KEY (`company_id`) REFERENCES `company` (`id`)), el error es causado porque existe una clave foránea que hace referencia a la tabla Company y no existen los datos introducidos en las demás tablas.



Para solucionar este problema hay que insertar un registro nuevo en credit_card con el id CcU-9999 ya que no existe.

```
92 • INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES
93   ('CcU-9999', 'TR3576817630138661950312 21', '5424646668135533', '3585', '458', '2023-10-31');
94
95 • select * from credit_card
96   where id = "CcU-9999";
97
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	iban	pan	pin	cvv	expiring_date
CcU-9999	TR3576817630138661950312 21	5424646668135533	3585	458	2023-10-31
NULL	NULL	NULL	NULL	NULL	NULL

credit_card 19

Output

Action Output

#	Time	Action	Message
1	16:47:56	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-9999', 'TR357681763013866...	1 row(s) affected
2	16:48:11	select * from credit_card where id = "CcU-9999"	1 row(s) returned

También hay que agregar un nuevo registro en Company con el id b-9999 ya que tampoco existe.

```
5058 • INSERT INTO company (id, company_name, phone, email, country, website) VALUES
5059   ('b-9999', 'Garden Smith Corp.', '07 45 87 32 28', 'garden.smith@protonmail.couk', 'United States', 'https://gardensmith.com');
5060
5061 • select * from company
5062   where id = "b-9999";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	company_name	phone	email	country	website
b-9999	Garden Smith Corp.	07 45 87 32 28	garden.smith@protonmail.couk	United States	https://gardensmith.com
NULL	NULL	NULL	NULL	NULL	NULL

company 10

Output

Action Output

#	Time	Action	Message
16	12:15:15	INSERT INTO company (id, company_name, phone, email, country, website) VALUES ('b-9999', 'Garden S...	1 row(s) affected
17	12:15:26	select * from company where id = "b-9999"	1 row(s) returned

Una vez añadidos los nuevos registros en las tablas Company y credit_card se podrá agregar el registro nuevo en transaction.

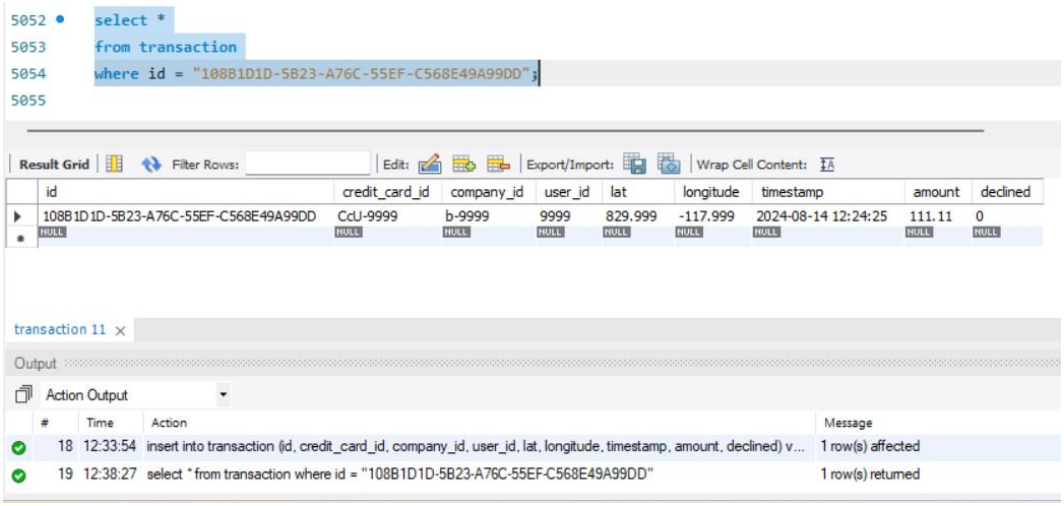
```
104 • insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) values
105   ("108B1D1D-5B23-A76C-55EF-C568E49A99DD", "CcU-9999", "b-9999", "9999", "829.999", "-117.999", "2024-08-14 12:24:25", "111.11", "0");
```

Output

Action Output

#	Time	Action	Message
1	16:56:14	insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) value...	1 row(s) affected

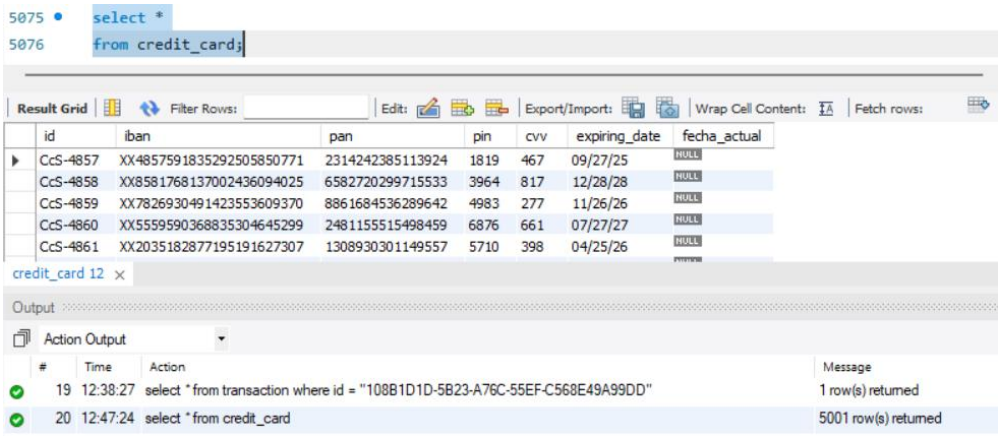
Una vez realizado lo anterior, hay que comprobar si existe algún registro con el indentificador id de la tabla anterior, se podrá observar que aparecen los datos registrados anteriormente.



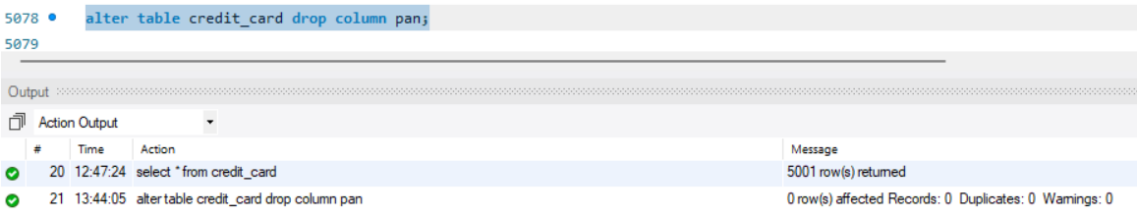
Ejercicio 4

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

Para este apartado primero se hace una selección de todo para ver que la columna pan existe en la tabla credit_card.



Para eliminar el campo pan será necesario utilizar alter table para modificar la tabla, definir el nombre de la tabla en la que queremos eliminar (credit_card), con drop column se podrá eliminar la columna que sea necesaria, en este caso pan.



Si se vuelve a verificar si sigue existiendo el campo pan en la tabla credit_card se podrá comprobar que ya no existe (también se puede comprobar con describe credit_card).

```
118 •  select *
119    from credit_card;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

id	iban	pin	cvv	expiring_date
CcS-4857	XX485759183529250585077 1	1819	467	2025-09-30
CcS-4858	XX858176813700243609402 5	3964	817	2028-12-31
CcS-4859	XX782693049142355360937 0	4983	277	2026-11-30
CcS-4860	XX555959036883530464529 9	6876	661	2027-07-31
CcS-4861	XX203518287719519162730 7	5710	398	2026-04-30
CcS-4862	XX477472146246364540975 8	4042	174	2026-11-30
CcS-4863	XX147682966424504620711 1	5969	449	2029-12-31
CcS-4864	XX838029889338573119615 9	8481	139	2026-02-28

credit_card 35 ×

Output

Action Output

#	Time	Action	Message
1	17:42:04	select * from credit_card	5001 row(s) returned

Nivel 2

Ejercicio 1

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

Para este apartado seleccionaremos todo de la tabla transaction y se pondrá un filtro para buscar el registro en el campo id y comprobar que existe.

```
5086 •  select *
5087    from transaction
5088    where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD";
5089
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	438	41.5972	12.2218	2016-12-21 20:07:18	155.63	0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

transaction 14 ×

Output

Action Output

#	Time	Action	Message
22	13:44:55	select * from credit_card	5001 row(s) returned
23	13:52:11	select * from transaction where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD"	1 row(s) returned

Una vez verificado, será necesario utilizar delete from, el nombre de la tabla (transaction) y un filtro (where) para buscar el registro del campo id, de esta manera, se borrara el registro que coincida con el que se ha puesto en el filtro.

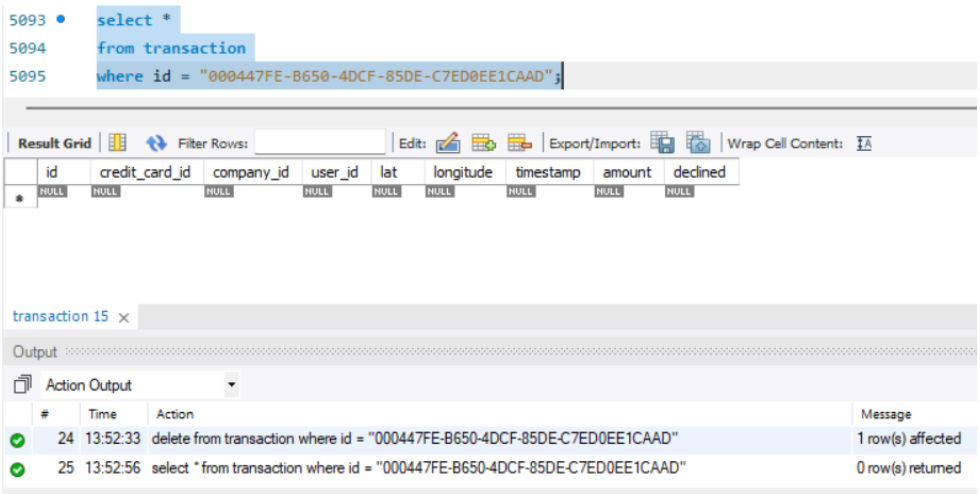
```
5090 •  delete from transaction
5091    where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD";
```

Output

Action Output

#	Time	Action	Message
23	13:52:11	select * from transaction where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD"	1 row(s) returned
24	13:52:33	delete from transaction where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD"	1 row(s) affected

Al volver a ejecutar la query que se utilizó para comprobar si el registro existía se podrá observar que ya no existe y ha sido eliminado correctamente.



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 necesario que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Media de compra realizada por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

Para realizar este apartado será necesario utilizar create view para crear la vista, ponerle un nombre (en este caso VistaMarketing) y hacer la query necesaria, la cual tiene la selección de los campos requeridos por la sección de marketing (nombre de la compañis, teléfono de contacto, país de residencia, media de compras realizadas por cada compañía) renombrados para mejor legibilidad, un join entre la tabla company y la tabla transaction, esta todo agrupado por el nombre de la compañía, el teléfono y el país, por último, se ordena de mayor a menor por la media del campo amount, redondeada a 2 decimales a la que se ha renombrado como mediaCompras.

Para ver los resultados de la vista será necesario seleccionar todo y en este caso en el from en vez de poner una tabla se pondrá el nombre de la vista.

```
177 •  select * from vistamarketing;
```

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

nombreCompañia	telefonoContacto	paisResidencia	mediaCompras
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

stamarketing 1 ×

Output

Action Output

#	Time	Action	Message
1	16:10:52	create view VistaMarketing as select company_name as nombreCompañia, phone as telefonoContacto, coun...	0 row(s) affected
2	16:12:17	select * from vistamarketing	101 row(s) returned

Ejercicio 3

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

Para realizar este apartado será necesario seleccionar todo para ver toda la información de la vista, y un filtro en el que se especificará que el país (paisResidencia, porque esta renombrado en la vista) sea Alemania, de esta manera, el resultado será toda la información de las empresas ubicadas en Alemania.

```
181 •  select *
182    from vistamarketing
183    where paisResidencia = "Germany";
```

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

nombreCompañia	telefonoContacto	paisResidencia	mediaCompras
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
Augue Foundation	06 88 43 15 63	Germany	253.51
Aliquam PC	01 45 73 52 16	Germany	253.14

vistamarketing 2 ×

Output

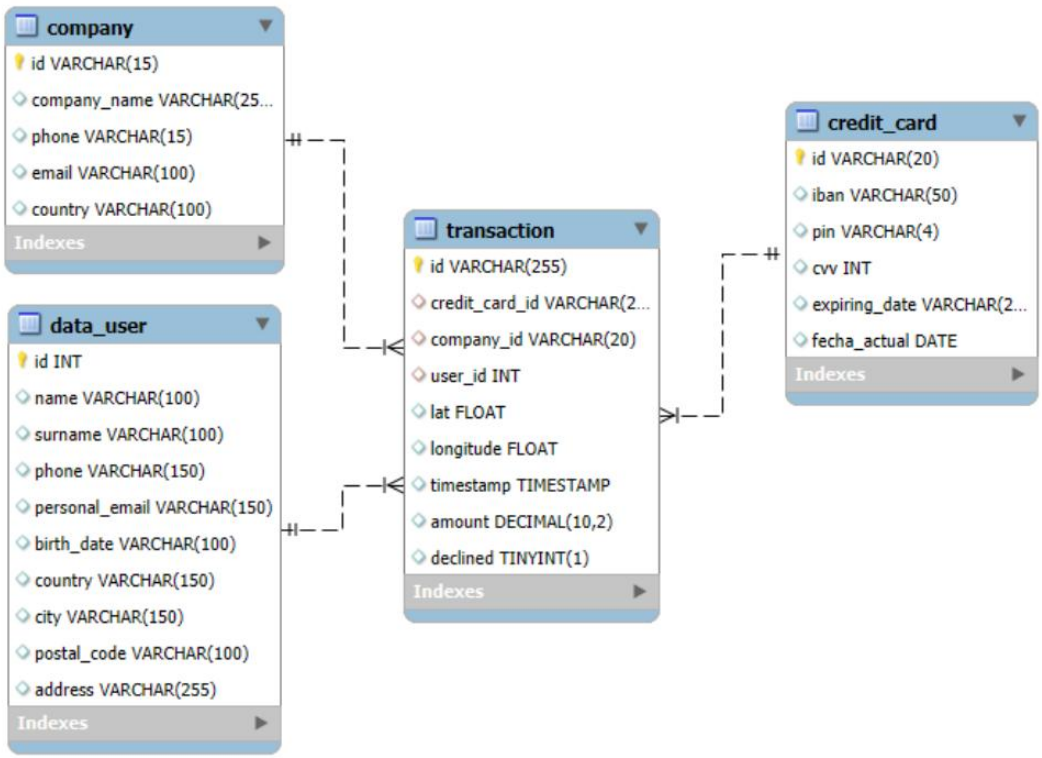
Action Output

#	Time	Action	Message
1	16:16:28	select * from vistamarketing where paisResidencia = "Germany"	8 row(s) returned

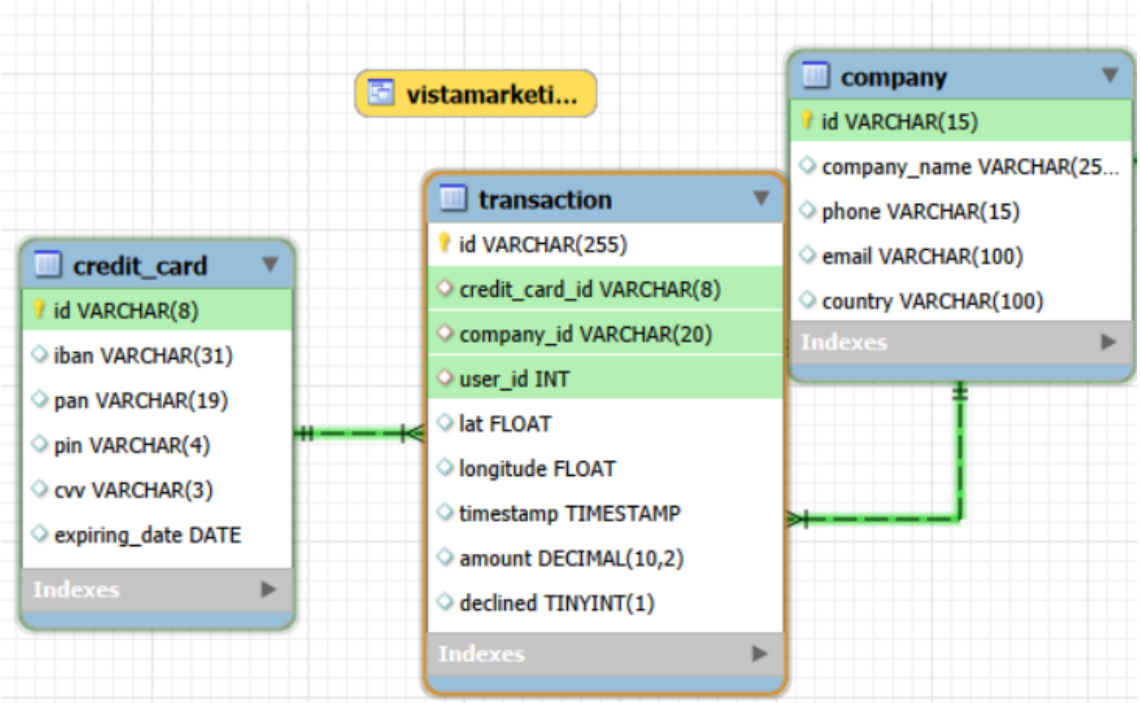
Nivel 3

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:



El diagrama inicial estaría así (hay algunos campos que coinciden con la tabla anterior porque es hay cosas corregidas y ya estaban los campos redefinidos)



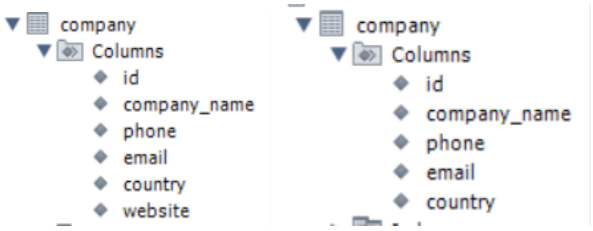
Primero se eliminará la columna website de la tabla Company, para ello será necesario utilizar alter table, el nombre de la tabla de la que queremos borrar la columna (company) y drop column con el nombre de la columna.

```
5064 •  alter table company
5065    drop column website;
5066
```

Output

Action Output

#	Time	Action	Message
29	18:35:33	DROP TABLE `transactions`.`user`	0 row(s) affected
30	18:39:47	alter table company drop column website	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0



Luego, en la tabla credit_card modificamos los siguientes campos:

Id varchar(8) a id varchar(20):

Para poder modificarlo hay que borrar la clave foránea de transaction, si no, dará error, para ello ejecutamos las siguientes querys, primero borramos la clave foránea, y luego modificamos el dato

```
5067 •  ALTER TABLE transaction
5068    DROP FOREIGN KEY fk_credit_card_id;
5069
5070 •  alter table credit_card
5071    modify id varchar(20);
5072
```

Output

Action Output

#	Time	Action	Message
34	18:55:24	ALTER TABLE transaction DROP FOREIGN KEY fk_credit_card_id	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
35	18:55:33	alter table credit_card modify id varchar(20)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0

Iban varchar(31) a iban varchar(50)

```
170 •  alter table credit_card
171    modify iban varchar(50);
```

Output

Action Output

#	Time	Action	Message
1	17:30:30	alter table credit_card modify id varchar(20)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	17:31:55	alter table credit_card modify iban varchar(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Pin varchar(19) a pin varchar(4)

```
173 •  alter table credit_card
174    modify pin varchar(4);
```

Output

Action Output

#	Time	Action	Message
1	17:30:30	alter table credit_card modify id varchar(20)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	17:31:55	alter table credit_card modify iban varchar(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	17:32:42	alter table credit_card modify pin varchar(4)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0

Cvv varchar(4) a cvv int

```
176 •  alter table credit_card
177    modify cvv int;
```

Output

Action Output

#	Time	Action	Message
1	17:30:30	alter table credit_card modify id varchar(20)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	17:31:55	alter table credit_card modify iban varchar(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	17:32:42	alter table credit_card modify pin varchar(4)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
4	17:33:19	alter table credit_card modify cvv int	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0

Expiring_date date a expiring_date varchar(255)

```
179 •  alter table credit_card
180    modify expiring_date varchar(255);
```

Output

Action Output

#	Time	Action	Message
1	17:36:15	alter table credit_card modify expiring_date varchar(255)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0

Se añade la columna fecha_actual

```
179 •  alter table credit_card
180    add fecha_actual date default (current_date);
```

Output

Action Output

#	Time	Action	Message
2	17:31:55	alter table credit_card modify iban varchar(50)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	17:32:42	alter table credit_card modify pin varchar(4)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
4	17:33:19	alter table credit_card modify cvv int	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
5	17:33:54	alter table credit_card add fecha_actual date default (current_date)	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0

Table: credit_card

Columns:

id	varchar(20) PK
iban	varchar(50)
pin	varchar(4)
cvv	int
expiring_date	varchar(255)
fecha_actual	date

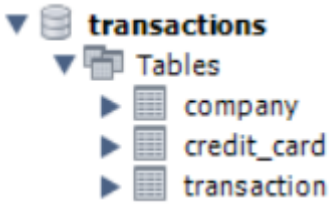
Ahora se crea la tabla user, para ello se utilizará la query del archivo estructura datos user.sql

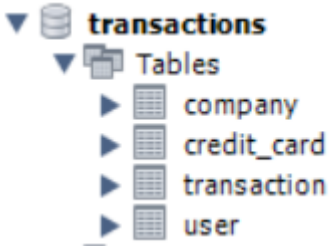
```
5085  CREATE TABLE IF NOT EXISTS user (
5086      id CHAR(10) PRIMARY KEY,
5087      name VARCHAR(100),
5088      surname VARCHAR(100),
5089      phone VARCHAR(150),
5090      email VARCHAR(150),
5091      birth_date VARCHAR(100),
5092      country VARCHAR(150),
5093      city VARCHAR(150),
5094      postal_code VARCHAR(100),
5095      address VARCHAR(255)
5096  );
5097
```

Output

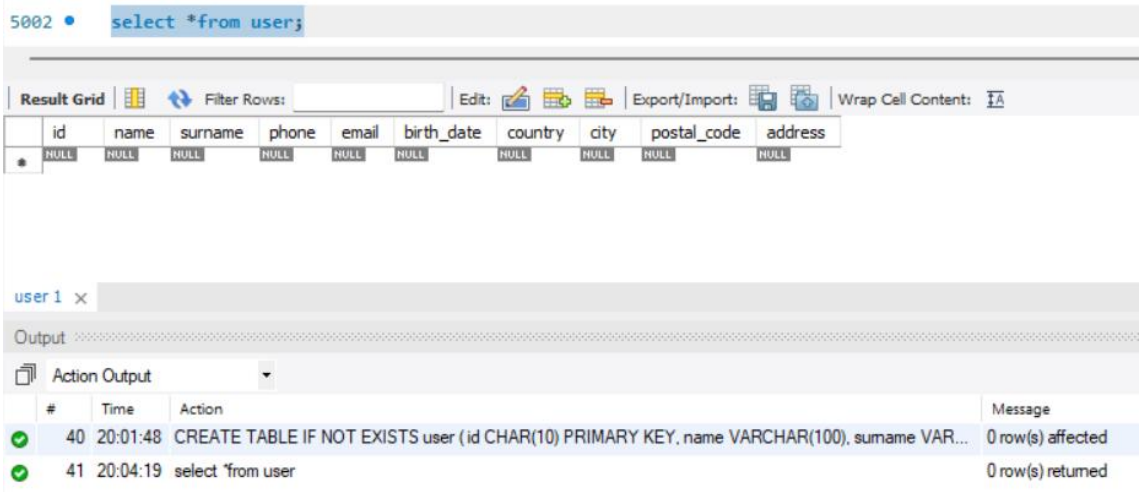
Action Output

#	Time	Action	Message
39	19:54:06	alter table credit_card modify fecha_actual date	5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
40	20:01:48	CREATE TABLE IF NOT EXISTS user (id CHAR(10) PRIMARY KEY, name VARCHAR(100), surname VAR...	0 row(s) affected

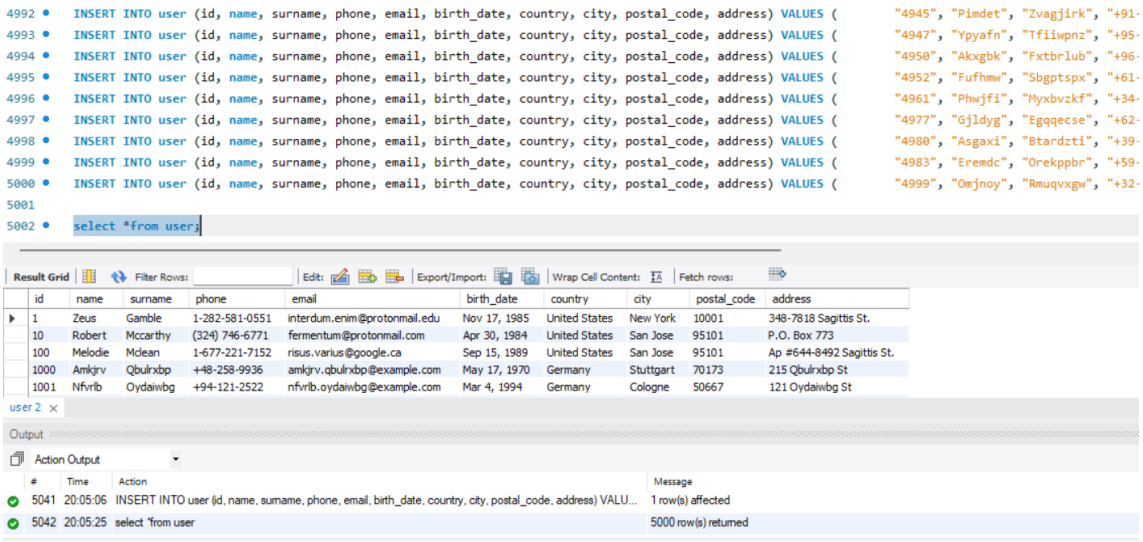




Se comprueba que la tabla user esta vacía.

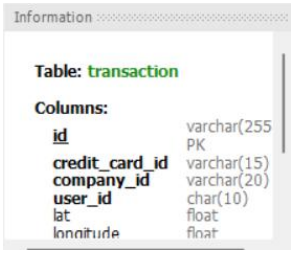


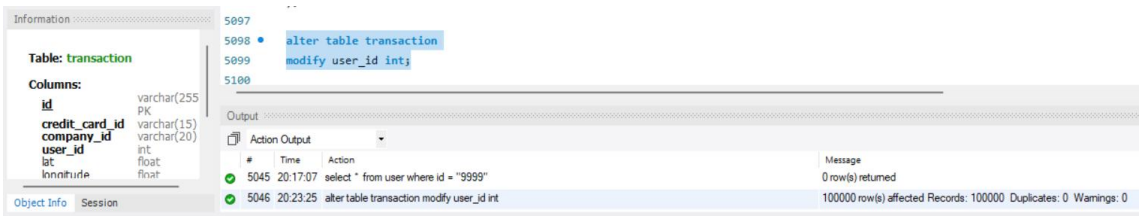
Se insertan las querys del archivo datos introducir sprint3 user.sql y se vuelve a comprobar si se han introducido todos los datos correctamente.



Ahora, se modificaran algunos valores de la tabla transaction y de la tabla user:

User_id char(10) a user_id int (tabla transaction)

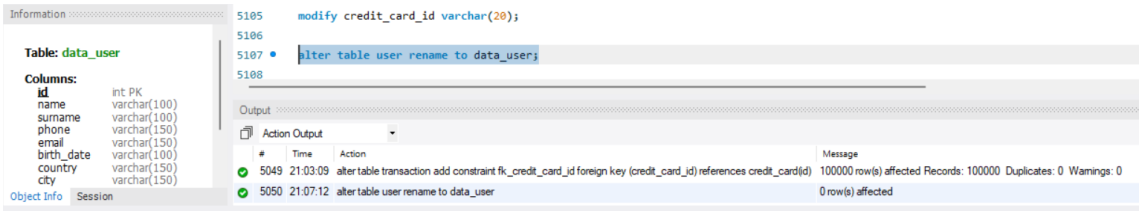




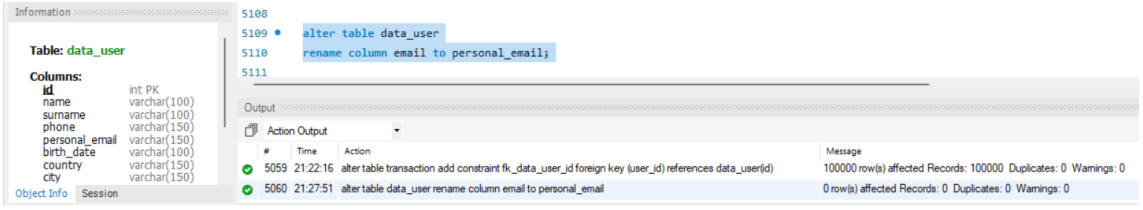
User_id char(10) a user_id int (tabla user)

Credit_card_id varchar(15) a credit_card_id varchar(20) (tabla transaction)

Renombramos la tabla user por data_user



Se renombra la columna email por personal_email (tabla data_user)

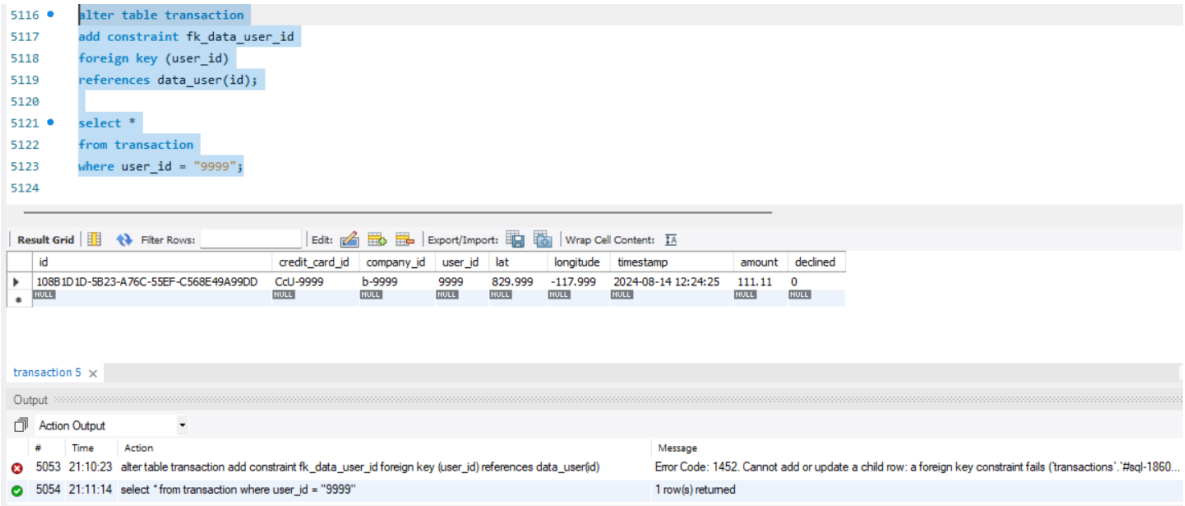


Se declara la clave foránea de la tabla credit_card en transaction

Se declara la clave foránea de la tabla data_user en transaction

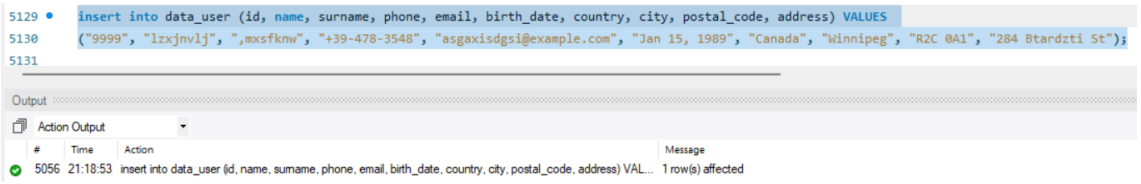
Da un error: (1452. Cannot add or update a child row: a foreign key constraint fails)

Este error ocurre porque hay datos que no coinciden en ambas tablas, en este caso el problema está en el registro que se añadió en el ejercicio 3 del nivel 1, se añadió un registro en el que el campo user_id tenía como usuario el 9999, el cual no existe en la tabla user

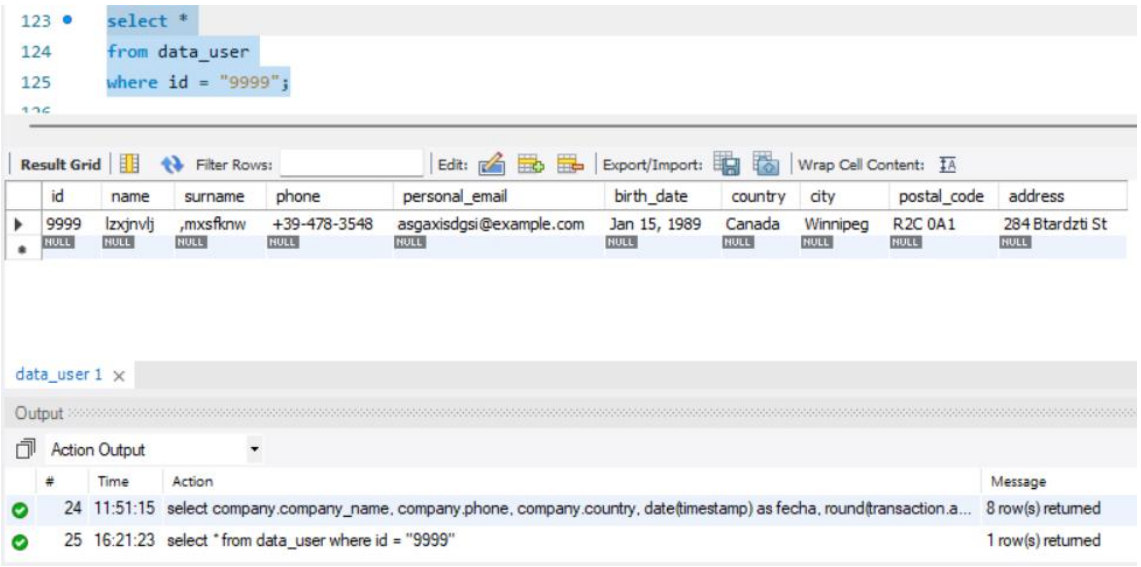


Se comprueba si existe el usuario en la tabla data_user

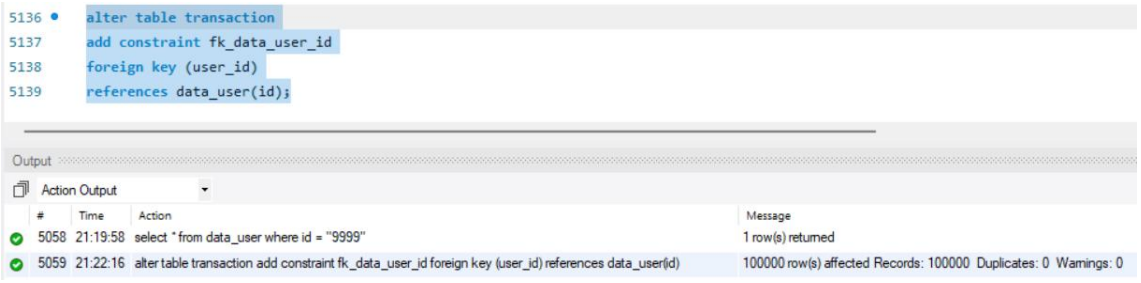
Como se aprecia en la imagen anterior, el usuario 9999 no existe en la tabla data_user, se crea con la siguiente query



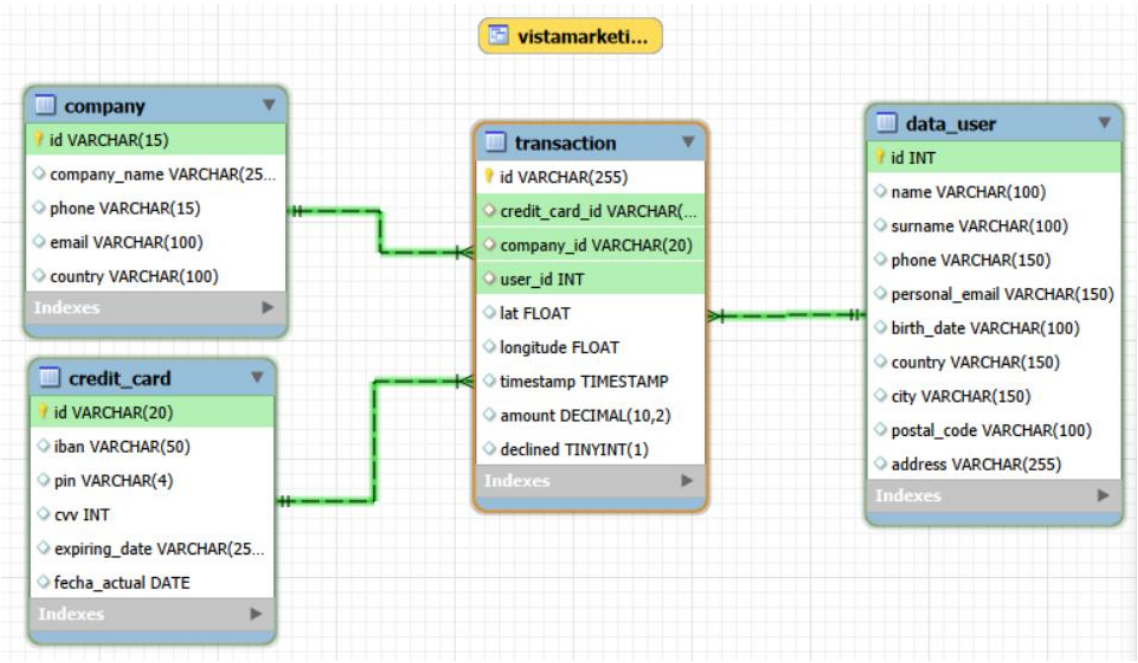
Al comprobar de nuevo el registro en la tabla data_user se podrá comprobar que ahora si existe



Si se ejecuta la query para declarar la clave foránea, ahora se podrá comprobar que se ejecuta sin problema y no da ningún error.



El diagrama entidad-relación final quedaría de la siguiente manera:



La relación entre la tabla data_user y transaction es 1-N de data_user a transaction, esto quiere decir que un usuario puede tener muchas transacciones, pero una transacción solo puede pertenecer a un usuario, en el caso de las otras dos tablas, se puede observar que también son relaciones 1-N tanto de Company hacia transaction y de credit_card hacia transaction, al igual que con data_user, esto quiere decir que una compañía puede tener muchas transaciones, pero una tranasccion solo puede pertenecer a una compañía y que una tarjeta de crédito puede tener muchas transacciones, pero una transacción solo puede pertenecer a una tarjeta de credito.

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

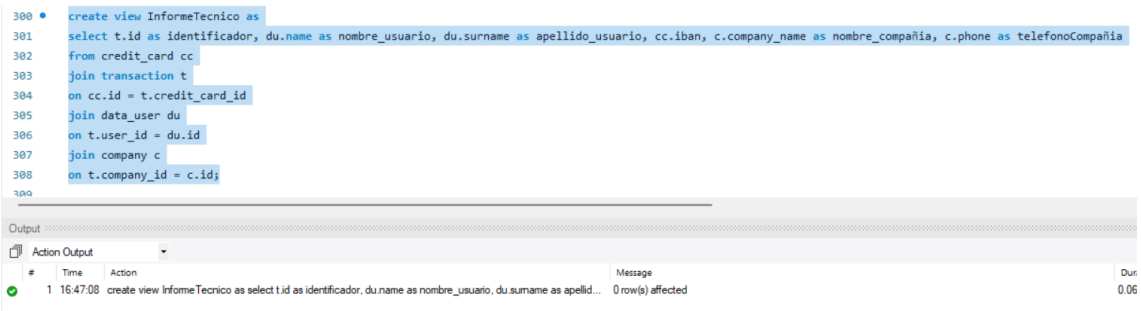
IBAN de la tarjeta de crédito usada.

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.

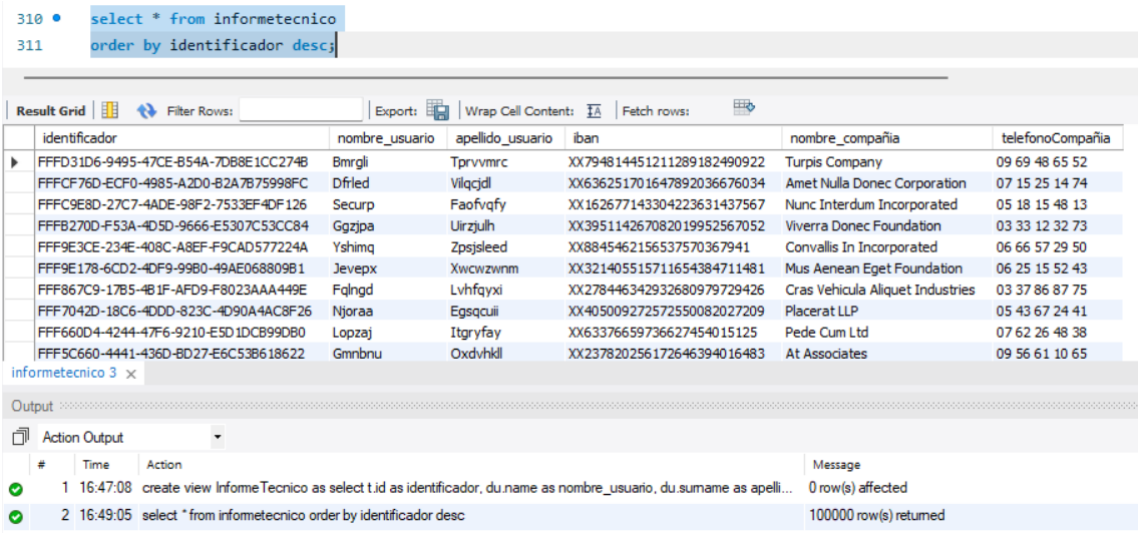
Muestra los resultados de la vista, ordena los resultados de forma descendente en función de la variable ID de transacción

Para este apartado se crea una vista con create view y se le pone InformeTecnico, en la selección se ponen todos los campos solicitados, como información relevante se ha añadido el teléfono de la compañia, se renombrar todo para que se mas legible y se unen la tabla credit_card, transaction, data_user y Company con join, luego se selecciona todo de la vista informetecnico y se ordena de mayor a menor por el campo identificador (que es el id de la tabla transaction).



```
300 • create view InformeTecnico as
301   select t.id as identificador, du.name as nombre_usuario, du.surname as apellido_usuario, cc.iban, c.company_name as nombre_compañia, c.phone as telefonoCompañia
302   from credit_card cc
303   join transaction t
304   on cc.id = t.credit_card_id
305   join data_user du
306   on t.user_id = du.id
307   join company c
308   on t.company_id = c.id;
```

```
Output
Action Output
#  Time      Action                                                                                  Message                Dur
1  16:47:08  create view InformeTecnico as select t.id as identificador, du.name as nombre_usuario, du.surname as apellid...  0 row(s) affected      0.06
```



```
310 • select * from informetecnico
311   order by identificador desc;
```

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

identificador	nombre_usuario	apellido_usuario	iban	nombre_compañia	telefonoCompañia
FFFD31D6-9495-47CE-B54A-7DB8E1CC274B	Bmrgli	Tprvvmrc	XX7948144512112891824909 22	Turpis Company	09 69 48 65 52
FFFCF76D-ECF0-4985-A2D0-B2A7B75998FC	Dfrled	Vilqcjdl	XX6362517016478920366760 34	Amet Nulla Donec Corporation	07 15 25 14 74
FFFC9E8D-27C7-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	XX1626771433042236314375 67	Nunc Interdum Incorporated	05 18 15 48 13
FFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzjpa	Uirzjulh	XX3951142670820199525670 52	Viverra Donec Foundation	03 33 12 32 73
FFF9E3CE-234E-408C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537570367941	Convallis In Incorporated	06 66 57 29 50
FFF9E178-6CD2-4DF9-99B0-49AE068809B1	Jevepx	Xwcwzwnm	XX3214055157116543847114 81	Mus Aenean Eget Foundation	06 25 15 52 43
FFF867C9-17B5-4B1F-AFD9-F8023AAA449E	Fqlngd	Lvhfqyxi	XX2784463429326809797294 26	Cras Vehicula Aliquet Industries	03 37 86 87 75
FFF7042D-18C6-4DDD-823C-4D90A4AC8F26	Njoraa	Egsqcuii	XX4050092725725500820272 09	Placerat LLP	05 43 67 24 41
FFF660D4-4244-47F6-9210-E5D1DCB99DB0	Lopzaj	Itgryfay	XX6337665973662745401 5125	Pede Cum Ltd	07 62 26 48 38
FFF5C660-4441-436D-BD27-E6C53B618622	Gmnbnu	Oxdvhkll	XX2378202561726463940 16483	At Associates	09 56 61 10 65

informetecnico 3

```
Output
Action Output
#  Time      Action                                                                                  Message
1  16:47:08  create view InformeTecnico as select t.id as identificador, du.name as nombre_usuario, du.surname as apelli...  0 row(s) affected
2  16:49:05  select * from informetecnico order by identificador desc                                100000 row(s) returned
```