

Sprint 3: Manipulación de tablas

Santiago Álvarez Salvado

Contenido

Nivel 1.....	3
Ejercicio 1	3
Ejercicio 2	9
Ejercicio 3	11
Ejercicio 4	13
Nivel 2.....	14
Ejercicio 1	14
Ejercicio 2	15
Ejercicio 3	16
Nivel 3.....	17
Ejercicio 1	17
Ejercicio 2	26

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.

The screenshot shows the MySQL Workbench interface. In the top pane, a code editor displays the SQL command to create the 'credit_card' table:

```
7 • ① create table if not exists credit_card (
8     id varchar(255) primary key not null,
9     iban varchar(255),
10    pan varchar(255),
11    pin varchar(255),
12    cvv varchar(255),
13    expiring_date varchar(255)
14 );
```

In the bottom pane, the 'Output' tab shows the execution results:

#	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;

The screenshot shows the MySQL Workbench interface. In the top pane, a code editor displays the SQL command to insert data into the 'credit_card' table:

```
1   |- Insertamos datos de credit_card
2 • ② INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2938', 'TR301950312213576817638661', '5424465566813633', '3257', '984', '10/30/22');
3 • ③ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2945', '0026854763748537475216568680', '5142423821948828', '9880', '887', '08/24/23');
4 • ④ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2952', '86451VQL52710525608255', '4556 453 55 5287', '4598', '438', '06/29/21');
5 • ⑤ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2959', 'CR724247724435841535', '372461377349375', '3583', '667', '02/24/23');
6 • ⑥ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2966', '8672LKTQ15627628377363', '448566 886747 7265', '4900', '130', '10/29/24');
7 • ⑦ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2973', 'PT87806228135092429456346', '544 58654 54343 384', '8760', '887', '01/30/25');
8 • ⑧ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2980', 'DE3924188183086277136', '402400 7145845969', '5875', '596', '07/24/22');
9 • ⑨ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2987', 'GE89681434837748781813', '3763 747687 76666', '2298', '797', '10/31/23');
10 • ⑩ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-2994', 'BH62714428368066765294', '344283273252593', '7545', '595', '02/28/22');
11 • ⑪ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3001', 'CY49687426654774581266832110', '511722 924833 2244', '9562', '867', '09/16/22');
12 • ⑫ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3008', 'LU507216693616119230', '4485744464433884', '1856', '740', '04/05/25');
13 • ⑬ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3015', 'PS119398216295715968342456821', '3784 662233 17389', '3246', '822', '01/31/22');
14 • ⑭ INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-3022', 'GT169516285056977423121857', '5164 1379 4842 3951', '5610', '342', '04/25/25');
```

In the bottom pane, the 'Output' tab shows the execution results:

#	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, XX0544818626973038...	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, XX95533893105308890...	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, XX529710930508771264...	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, XX158914070859480863...	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, XX9153964645611056...	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, XX2963930915871020...	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, XX7812588985195080...	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 x		
Output		
Action Output		
#	Time	Action
1	22:00:07	select max(length(id)) as longitud_id from credit_card
		Message 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 x		
Output		
Action Output		
#	Time	Action
1	22:00:07	select max(length(id)) as longitud_id from credit_card
2	22:00:30	select max(length(ibanc)) as longitud_ibanc from credit_card
		Message 1 row(s) returned 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 x		
Output		
Action Output		
#	Time	Action
1	22:00:07	select max(length(id)) as longitud_id from credit_card
2	22:00:30	select max(length(ibanc)) as longitud_ibanc from credit_card
3	22:00:52	select max(length(pan)) as longitud_pan from credit_card
		Message 1 row(s) returned 1 row(s) returned 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 x

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 x

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 x

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 quizás 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

```
30 •  describe credit_card;
```

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
iban	varchar(255)	YES		NULL	
pan	varchar(255)	YES		NULL	
pin	varchar(255)	YES		NULL	
cvv	varchar(255)	YES		NULL	
expiring_date	varchar(255)	YES		NULL	

Result 2 ×

Output

Action Output

#	Time	Action
1	11:39:51	describe credit_card

Message
6 row(s) returned

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

```
32 •  alter table credit_card
33     modify id varchar(8);
```

Output

Action Output

#	Time	Action
1	11:39:51	describe credit_card
2	11:41:41	alter table credit_card modify id varchar(8)

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

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

```
35 •  alter table credit_card
36     modify iban varchar(31);
```

Output

Action Output

#	Time	Action
2	11:41:41	alter table credit_card modify id varchar(8)
3	11:42:34	select max(length(iban)) as longitud_iban from credit_card
4	11:42:48	alter table credit_card modify iban varchar(31)

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


```
38 •  alter table credit_card
39     modify pan varchar(19);
```

Output

Action Output

#	Time	Action
3	11:42:34	select max(length(iban)) as longitud_iban from credit_card
4	11:42:48	alter table credit_card modify iban varchar(31)
5	11:44:16	select max(length(pan)) as longitud_pan from credit_card
6	11:44:31	alter table credit_card modify pan varchar(19)

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

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

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

Output:

#	Time	Action	Message
5	11:44:16	select max(length(pin)) as longitud_pin from credit_card	1 row(s) returned
6	11:44:31	alter table credit_card modify pin 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 ocurriría lo mismo que con el pin.

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

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 sería 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 último 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;
```

Output:

#	Time	Action	Message
1	13:13:10	UPDATE credit_card SET credit_card.expiring_date = LAST_DAY(STR_TO_DATE(credit_card.expiring...) limit 6000;	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  
1 13:17:57 altertable credit_card modify expiring_date date  
Message  
5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
```

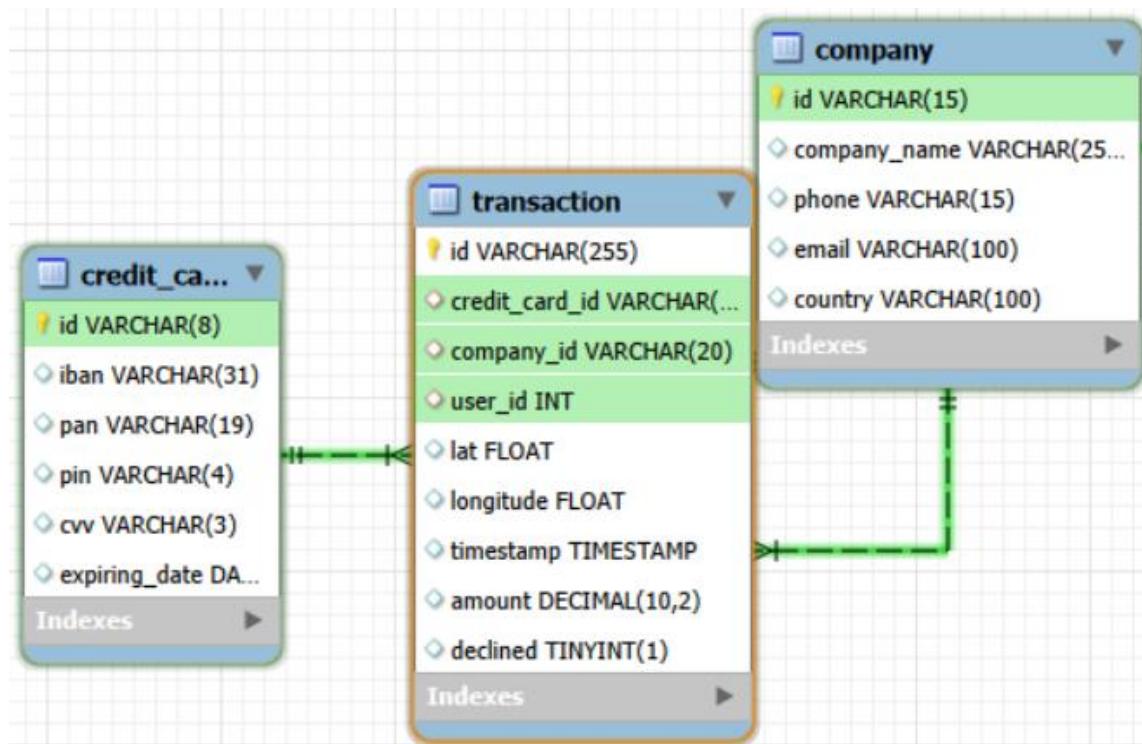
Se vuelve a comprobar los datos de la tabla para ver que se han modificado los campos correctamente.

```
57 • describe credit_cards;  
  
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 x  
  
Output :  
Action Output  
# Time Action  
1 13:17:57 altertable credit_card modify expiring_date date  
2 13:21:07 describe credit_card  
Message  
5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0  
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);  
  
Output :  
Action Output  
# Time Action  
1 16:35:10 altertable transaction add constraint fk_credit_cardid foreign key (credit_card_id) references credit_card(id)  
Message  
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 transaction, 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						
	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 x		
Output		
Action Output		
#	Time	Action
1	16:16:40	select * from credit_card where id = "CcU-2938"

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 = "TR323456312213576817699999"  
76   where id = "CcU-2938";  
77  
78 • select * from credit_card  
79   where id = "CcU-2938";
```

Result Grid							
		id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	TR323456312213576817699999	5424465566813633	3257	984	2022-10-31	
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

credit_card 9 ×						
Output						
Action Output					Message	
#	Time	Action				
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 = "TR323456312213576817699999" 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.

```
5049 •  insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) values
5050   ("108B1D1D-5B23-A76C-55EF-C568E49A99DD", "CcU-9999", "b-9999", "9999", "829.999", "-117.999", "111.11", "0");
```

Output			
#	Time	Action	Message
5087	17:29:44	insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) values ("108... Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('transactions`.`transaction`...	
5088	17:30:50	insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, amount, declined) values ("108... Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('transactions`.`transaction`...	

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', 'TR357681763013866195031221', '5424646668135533', '3585', '458', '2023-10-31');
94
95 • select * from credit_card
96 where id = "CcU-9999";
97

```

Result Grid						
	id	iban	pan	pin	cvv	expiring_date
▶	CcU-9999	TR357681763013866195031221	5424646668135533	3585	458	2023-10-31
*	HULL	HULL	HULL	HULL	HULL	HULL

credit_card 19 x						
Output						
Action Output						
#	Time	Action				Message
✓	1 16:47:56	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ('CcU-9999', 'TR357681763013866195031221', '5424646668135533', '3585', '458', '2023-10-31')				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.co.uk', 'United States', 'https://gardensmith.com');
5060
5061 • select * from company
5062 where id = "b-9999";

```

Result Grid						
	id	company_name	phone	email	country	website
▶	b-9999	Garden Smith Corp.	07 45 87 32 28	garden.smith@protonmail.co.uk	United States	https://gardensmith.com
*	HULL	HULL	HULL	HULL	HULL	HULL

company 10 x						
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...', '07 45 87 32 28', 'garden.smith@protonmail.co.uk', 'United States', 'https://gardensmith.com')				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     ("1088101D-5B23-A76C-55EF-C568E49A90D", "CcU-9999", "b-9999", "9999", "829.999", "-117.999", "2024-08-14 12:24:25", "111.11", "0");
106

```

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 identificador id de la tabla anterior, se podrá observar que aparecen los datos registrados anteriormente.

```
5052 • select *
5053   from transaction
5054   where id = "108B1D1D-5B23-A76C-55EF-C568E49A99DD";
5055

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content: 
id credit_card_id company_id user_id lat longitude timestamp amount declined
▶ 108B1D1D-5B23-A76C-55EF-C568E49A99DD CcU-9999 b-9999 9999 829,999 -117,999 2024-08-14 12:24:25 111.11 0
* NULL NULL
```

transaction 11 ×

Output

#	Time	Action	Message
18	12:33:54	insert into transaction (id, credit_card_id, company_id, user_id, lat, longitude, timestamp, amount, declined) v...	1 row(s) affected
19	12:38:27	select * from transaction where id = "108B1D1D-5B23-A76C-55EF-C568E49A99DD"	1 row(s) returned

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.

```
5075 • select *
5076   from credit_card;
```

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

id	iban	pan	pin	cvv	expiring_date	fecha_actual
CcS-4857	XX4857591835292505850771	2314242385113924	1819	467	09/27/25	NULL
CcS-4858	XX6581768137002436094025	6582720299715533	3964	817	12/28/28	NULL
CcS-4859	XX7826930491428553609370	8861684536289642	4983	277	11/26/26	NULL
CcS-4860	XX5559590368835304645299	2481155515498459	6876	661	07/27/27	NULL
CcS-4861	XX2035182877195191627307	1308930301149557	5710	398	04/25/26	NULL

credit_card 12 ×

Output

#	Time	Action	Message
19	12:38:27	select * from transaction where id = "108B1D1D-5B23-A76C-55EF-C568E49A99DD"	1 row(s) returned
20	12:47:24	select * from credit_card	5001 row(s) returned

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.

```
5078 • alter table credit_card drop column pan;
5079

Output
```

#	Time	Action	Message
20	12:47:24	select * from credit_card	5001 row(s) returned
21	13:44:05	alter table credit_card drop column pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

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

id	iban	pin	cvv	expiring_date
CcS-4857	XX485759183529250580771	1819	467	2025-09-30
CcS-4858	XX8581768137002436094025	3964	817	2028-12-31
CcS-4859	XX7826930491423553609370	4983	277	2026-11-30
CcS-4860	XX5559590368835304645299	6876	661	2027-07-31
CcS-4861	XX2035182877195191627307	5710	398	2026-04-30
CcS-4862	XX4774721462463645409758	4042	174	2026-11-30
CcS-4863	XX1476829664245046207111	5969	449	2029-12-31
CcS-4864	XX8380298893385731196159	8481	139	2026-02-28
CcS-4865	VV70850798061107580500	7847	0n3	2028-11-30

credit_card 35 x

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

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

transaction 14 x

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.

```
5093 • select *
5094   from transaction
5095  where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD";
```

	id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

transaction 15 x

Output :

Action Output
Time Action Message
24 13:52:33 delete from transaction where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD" 1 row(s) affected
25 13:52:56 select * from transaction where id = "000447FE-B650-4DCF-85DE-C7ED0EE1CAAD" 0 row(s) returned

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 compañía, 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.

```
168 • create view VistaMarketing as
169   select company_name as nombreCompañia, phone as telefonoContacto, country as paisResidencia, round(avg(amount), 2) as mediaCompras
170   from company
171   join transaction
172   on company.id = transaction.company_id
173   group by company_name, phone, country
174   order by mediaCompras desc;
175
```

Output :

Action Output
Time Action Message
1 16:10:52 create view VistaMarketing as select company_name as nombreCompañia, phone as telefonoContacto, count... 0 row(s) affected

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;
```

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

vistamarketing 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";
```

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 :

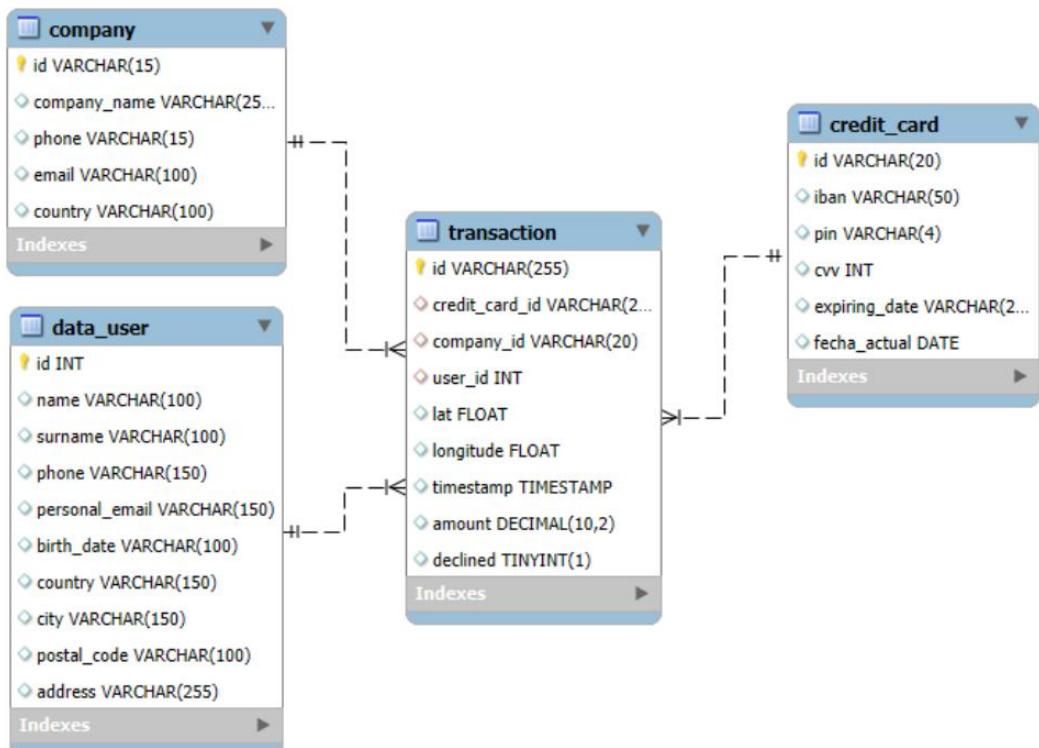
Action Output

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

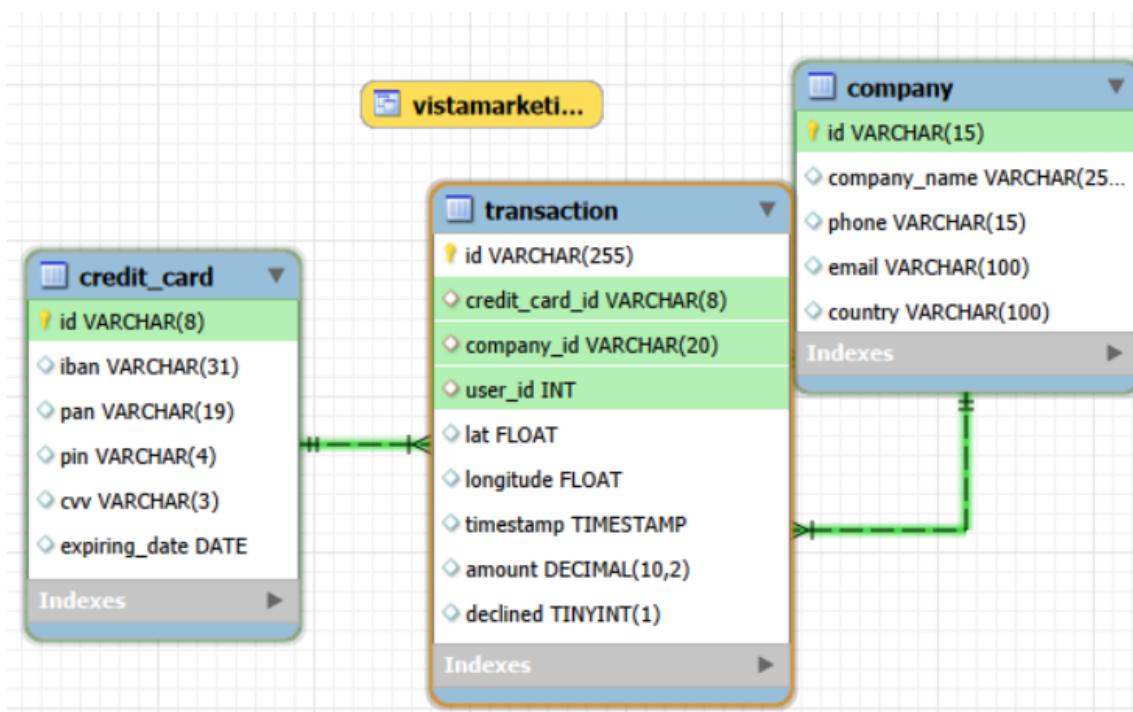
Nivel 3

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
29 18:35:33 DROP TABLE `transactions`.`user`
0 row(s) affected
30 18:39:47 altertable company drop column website
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

▼ company          ▼ company
  ▼ Columns        ▼ Columns
    ◆ id             ◆ id
    ◆ company_name   ◆ company_name
    ◆ phone           ◆ phone
    ◆ email            ◆ email
    ◆ country          ◆ country
    ◆ website

```

Luego, en la tabla credit_card modificamos los siguientes campos:

Table: credit_card

Columns:

<u>id</u>	varchar(8) PK
<u>iban</u>	varchar(31)
<u>pin</u>	varchar(19)
<u>cvv</u>	varchar(4)
<u>expiring_date</u>	date

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
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 altertable 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  
1 17:30:30 alter table credit_card modify id varchar(20)  
2 17:31:55 alter table credit_card modify iban varchar(50)  
Message  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
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  
1 17:30:30 alter table credit_card modify id varchar(20)  
2 17:31:55 alter table credit_card modify iban varchar(50)  
3 17:32:42 alter table credit_card modify pin varchar(4)  
Message  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
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  
1 17:30:30 alter table credit_card modify id varchar(20)  
2 17:31:55 alter table credit_card modify iban varchar(50)  
3 17:32:42 alter table credit_card modify pin varchar(4)  
4 17:33:19 alter table credit_card modify cvv int  
Message  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0  
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  
1 17:36:15 alter table credit_card modify expiring_date varchar(255)  
Message  
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  
2 17:31:55 alter table credit_card modify iban varchar(50)  
3 17:32:42 alter table credit_card modify pin varchar(4)  
4 17:33:19 alter table credit_card modify cvv int  
5 17:33:54 alter table credit_card add fecha_actual date default (current_date)  
Message  
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0  
5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0  
5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0  
5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
```

Table: credit_card

Columns:

<u>id</u>	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
1 39 19:54:06 alter table credit_card modify fecha_actual date
2 40 20:01:48 CREATE TABLE IF NOT EXISTS user (id CHAR(10) PRIMARY KEY, name VARCHAR(100), surname VAR... 0 row(s) affected
Message
5001 row(s) affected Records: 5001 Duplicates: 0 Warnings: 0
```



Se comprueba que la tabla user esta vacía.

5002 • select *from user;

Result Grid

	id	name	surname	phone	email	birth_date	country	city	postal_code	address
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

user 1 x

Output

Action Output

#	Time	Action	Message
40	20:01:48	CREATE TABLE IF NOT EXISTS user (id CHAR(10) PRIMARY KEY, name VARCHAR(100), surname VAR...	0 row(s) affected
41	20:04:19	select *from user	0 row(s) returned

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

4992 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4993 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4994 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4995 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4996 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4997 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4998 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

4999 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

5000 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (

5001

5002 • select *from user;

Result Grid

	id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York	10001	348-7818 Sagittis St.	
10	Robert	McCarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	San Jose	95101	P.O. Box 773	
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca	Sep 15, 1989	United States	San Jose	95101	Ap #644-8492 Sagittis St.	
1000	Amikry	Qbulrbp	+48-258-9936	amikry.qbulrbp@example.com	May 17, 1970	Germany	Stuttgart	70173	215 Qbulrbp St	
1001	Nfvrbl	Oydaiwbg	+94-121-2522	nfvrbl.oydaiwbg@example.com	Mar 4, 1994	Germany	Cologne	50667	121 Oydaiwbg St	

user 2 x

Output

Action Output

#	Time	Action	Message
5041	20:05:06	INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALU...	1 row(s) affected
5042	20:05:25	select *from user	5000 row(s) returned

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

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

Information

Table: transaction

Columns:

id	varchar(255)
credit_card_id	varchar(15)
company_id	varchar(20)
user_id	char(10)
lat	float
longitude	float

Information

Table: transaction

Columns:

id	varchar(255)
----	--------------

5097
5098 • alter table transaction
5099 modify user_id int;
5100

Output

Action Output

#	Time	Action	Message
5045	20:17:07	select *from user where id = "9999"	0 row(s) returned
5046	20:23:25	alter table transaction modify user_id int	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

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

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' panel displays the 'Table: user'. It has 5 columns: id (char(10) PK), name (varchar(100)), surname (varchar(100)), phone (varchar(150)), and email (varchar(150)). On the right, the 'Output' panel shows the results of the following SQL command:

```
5100
5101 • alter table user
5102   modify id int;
5103
```

The 'Action Output' section shows two rows of data:

#	Time	Action
5046	20:23:25	alter table transaction modify user_id int
5047	20:26:03	alter table user modify id int

The 'Message' section indicates: 100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0.

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

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' panel displays the 'Table: transaction'. It has 7 columns: id (varchar(255) PK), credit_card_id (varchar(20)), company_id (varchar(20)), user_id (int), lat (float), longitude (float), and timestamp (timestamp). On the right, the 'Output' panel shows the results of the following SQL command:

```
5103
5104 • alter table transaction
5105   modify credit_card_id varchar(20);
5106
```

The 'Action Output' section shows two rows of data:

#	Time	Action
5047	20:26:03	alter table user modify id int
5048	20:59:41	alter table transaction modify credit_card_id varchar(20)

The 'Message' section indicates: 5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0.

Renombramos la tabla user por data_user

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' panel displays the 'Table: data_user'. It has 7 columns: id (int PK), name (varchar(100)), surname (varchar(100)), phone (varchar(150)), email (varchar(150)), birth_date (varchar(100)), country (varchar(150)), and city (varchar(150)). On the right, the 'Output' panel shows the results of the following SQL command:

```
5105   modify credit_card_id varchar(20);
5106
5107 • alter table user rename to data_user;
5108
```

The 'Action Output' section shows two rows of data:

#	Time	Action
5049	21:03:09	alter table transaction add constraint fk_credit_card_id foreign key (credit_card_id) references credit_card(id)
5050	21:07:12	alter table user rename to data_user

The 'Message' section indicates: 100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0.

Se renombra la columna email por personal_email (tabla data_user)

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' panel displays the 'Table: data_user'. It has 7 columns: id (int PK), name (varchar(100)), surname (varchar(100)), phone (varchar(150)), personal_email (varchar(150)), birth_date (varchar(100)), country (varchar(150)), and city (varchar(150)). On the right, the 'Output' panel shows the results of the following SQL command:

```
5108
5109 • alter table data_user
5110   rename column email to personal_email;
5111
```

The 'Action Output' section shows two rows of data:

#	Time	Action
5059	21:22:16	alter table transaction add constraint fk_data_user_id foreign key (user_id) references data_user(id)
5060	21:27:51	alter table data_user rename column email to personal_email

The 'Message' section indicates: 100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0.

Se declara la clave foránea de la tabla credit_card en transaction

The screenshot shows the MySQL Workbench interface. On the left, the 'Information' panel displays the 'Table: transaction'. It has 7 columns: id (varchar(255) PK), credit_card_id (varchar(20)), company_id (varchar(20)), user_id (int), lat (float), longitude (float), and timestamp (timestamp). On the right, the 'Output' panel shows the results of the following SQL command:

```
5107 • alter table transaction
5108   add constraint fk_credit_card_id
5109     foreign key (credit_card_id)
5110       references credit_card(id);
5111
```

The 'Action Output' section shows two rows of data:

#	Time	Action
5048	20:59:41	alter table transaction modify credit_card_id varchar(20)
5049	21:03:09	alter table transaction add constraint fk_credit_card_id foreign key (credit_card_id) references credit_card(id)

The 'Message' section indicates: 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0.

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

```
5116 • alter table transaction
5117   add constraint fk_data_user_id
5118     foreign key (user_id)
5119       references data_user(id);
5120
5121 • select *
5122   from transaction
5123   where user_id = "9999";
```

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

id	credit_card_id	company_id	user_id	lat	longitude	timestamp	amount	declined
10881D1D-5B23-A76C-55EF-C568E49A99DD	CdU-9999	b-9999	9999	829.999	-117.999	2024-08-14 12:24:25	111.11	0
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

transaction 5 x

Output

Action Output

#	Time	Action
✖	5053 21:10:23	alter table transaction add constraint fk_data_user_id foreign key (user_id) references data_user(id)
✔	5054 21:11:14	select * from transaction where user_id = "9999"

Message

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (transactions`.`#sql-1860...)

1 row(s) returned

Se comprueba si existe el usuario en la tabla data_user

```
5125 • select *
5126   from data_user
5127   where id = "9999";
```

5128

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

id	name	surname	phone	email	birth_date	country	city	postal_code	address
*	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL	HULL

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

```
5129 • insert into data_user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES
5130   ("9999", "lzxjnvlij", ",mxfknew", "+39-478-3548", "asgaxisdgs@example.com", "Jan 15, 1989", "Canada", "Winnipeg", "R2C 0A1", "284 Btardzti St");
5131
```

Output

Action Output

#	Time	Action
✔	5056 21:18:53	insert into data_user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VAL... 1 row(s) affected

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

```
123 • select *
124   from data_user
125   where id = "9999";
126
```

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

id	name	surname	phone	personal_email	birth_date	country	city	postal_code	address
9999	Izxjnvlj	mxsflkw	+39-478-3548	asgaxisdgs@example.com	Jan 15, 1989	Canada	Winnipeg	R2C 0A1	284 Btardzti St
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

data_user 1 ×

Output

Action Output

#	Time	Action	Message
24	11:51:15	select company.company_name, company.phone, company.country, date(timestamp) as fecha, round(transaction.a...	8 row(s) returned
25	16:21:23	select * from data_user where id = "9999"	1 row(s) returned

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

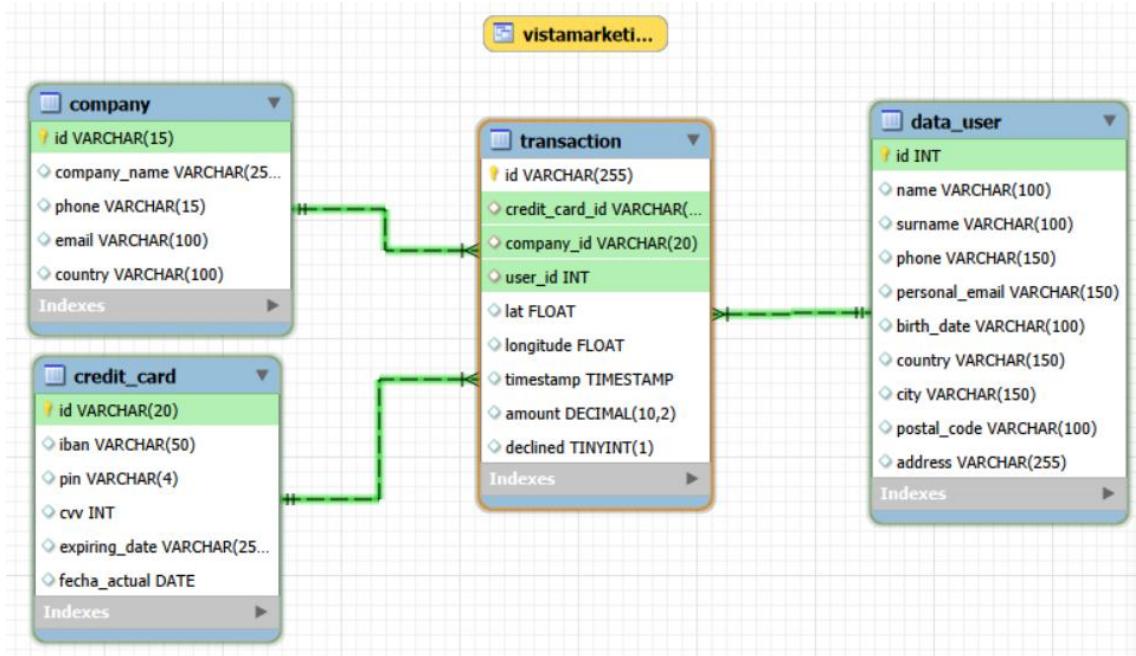
```
5136 • alter table transaction
5137   add constraint fk_data_user_id
5138   foreign key (user_id)
5139   references data_user(id);
```

Output

Action Output

#	Time	Action	Message
5058	21:19:58	select * from data_user where id = "9999"	1 row(s) returned
5059	21:22:16	alter table transaction add constraint fk_data_user_id foreign key (user_id) references data_user(id)	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

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 transacciones, pero una transacción 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 crédito.

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 compañía, se renombran todo para que sea 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
309
310 •  select * from informetecnico
311   order by identificador desc;
```

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 apellido_usuario, cc.iban, c.company_name as nombre_compañia, c.phone as telefonoCompañia from credit_card cc join transaction t on cc.id = t.credit_card_id join data_user du on t.user_id = du.id join company c on t.company_id = c.id	0 row(s) affected	0.06

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	Tprvmmrc	XX794814451211289182490922	Turpis Company	09 69 48 65 52
FFF9F76D-0CF0-4985-A2D0-B2A7B75998FC	Dfried	Vilqcdl	XX636251701647892036676034	Amet Nulla Donec Corporation	07 15 25 14 74
FFFC9E8D-2C7C-4ADE-98F2-7533EF4DF126	Securp	Faofvqfy	XX162677143304223631437567	Nunc Interdum Incorporated	05 18 15 48 13
FFFFB270D-F53A-4D5D-9666-E5307C53CC84	Ggzjpa	Uirzjulh	XX395114267082019952567052	Viverra Donec Foundation	03 33 12 32 73
FFF9E3CE-4234-E08C-A8EF-F9CAD577224A	Yshimq	Zpsjsleed	XX8845462156537507367941	Convallis In Incorporated	06 66 57 29 50
FFF9E178-6CD2-40F9-9980-49AE06880981	Jevepx	Xwcwzwnm	XX321405515711654384711481	Mus Aenean Eget Foundation	06 25 15 52 43
FFF867C9-17B5-4B1F-AD9-F8023AAA449E	Folngd	Lvhfqxyi	XX278446342932680979729426	Cras Vehicula Aliquet Industries	03 37 86 87 75
FFF7042D-18C6-4DD-823C-4D90A4AC8F26	Njoraa	Egsquii	XX405009272572550082027209	Placerat LLP	05 43 67 24 41
FFF660D4-4244-47F6-9210-ESD1DCB99D80	Lopzaj	Itgryfay	XX63376659736627454015125	Pede Cum Ltd	07 62 26 48 38
FFF5C660-4411-436D-8D27-E6C53B618622	Gmnbru	Oxdvhkll	XX237820256172646394016483	At Associates	09 56 61 10 65

informetecnico 3

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 apellido_usuario, cc.iban, c.company_name as nombre_compañia, c.phone as telefonoCompañia from credit_card cc join transaction t on cc.id = t.credit_card_id join data_user du on t.user_id = du.id join company c on t.company_id = c.id	0 row(s) affected
2	16:49:05	select * from informetecnico order by identificador desc	100000 row(s) returned