

## Tarea S3.01. Manipulación de tablas

<b>1. Nivel 1.....</b>	<b>1</b>
1.1. Ejercicio 1.....	1
• Crear de la tabla.....	1
• Insertar datos.....	2
• Creación de las relaciones.....	5
• Diagrama ER.....	6
1.2. Ejercicio 2.....	7
• Modificar de datos.....	7
• Comprobar cambio realizado.....	7
1.3. Ejercicio 3.....	8
• Insertar nuevo registro en transaction.....	9
1.4. Ejercicio 4.....	9
• Eliminar columna.....	9
• Comprobar cambio realizado.....	10
<b>2. Nivel 2.....</b>	<b>11</b>
2.1. Ejercicio 1.....	11
• Eliminar registro.....	11
• Comprobar cambio realizado.....	11
2.2. Ejercicio 2.....	12
• Crear la vista.....	12
• Mostrar resultados de la vista.....	13
2.3. Ejercicio 3.....	14
<b>3. Nivel 3.....</b>	<b>15</b>
3.1. Ejercicio 1.....	15
• Comparación con modelo anterior y definición de modificaciones necesarias por tablas.....	16
• Creación de tabla user, ingesta de datos y relación con tabla transaction.....	17
• Modificaciones en tablas company y credit_card.....	20
• Generación del Modelo ER.....	22
3.2. Ejercicio 2.....	22
• Crear la vista.....	23
• Mostrar resultados de la vista.....	24

## 1. Nivel 1

### 1.1. Ejercicio 1

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.

- Crear de la tabla

#### Consideraciones iniciales

- Identificador único de la tarjeta
- Nombre del titular de la tarjeta
- Código IBAN único por tarjeta
- Código PIN secreto de 4 dígitos
- Número principal de la tarjeta
- Código de seguridad de la tarjeta
- Fecha de vencimiento de la tarjeta en formato

```
CREATE TABLE IF NOT EXISTS credit_card (  
    id VARCHAR(15) PRIMARY KEY,  
    -- nombre VARCHAR(100) NOT NULL,  
    iban VARCHAR(34) NOT NULL UNIQUE,  
    pan VARCHAR(16) NOT NULL UNIQUE,  
    pin VARCHAR(4) NOT NULL,  
    cvv VARCHAR(4) NOT NULL,  
    expiring_date DATE NOT NULL);
```

#### Explicación:

- Para diseñar la tabla credit\_card, se tuvieron en cuenta los datos del archivo "datos\_introducir\_credit"
- Inicialmente fue considerado el campo **nombre**, pero como no estaba presente en los datos del archivo, quedó comentado en el código.
- Se utilizó VARCHAR para **id**, **iban**, **pan**, **pin** y **cvv** debido a su naturaleza alfanumérica.
- **expiring\_date** se estableció como DATE para facilitar la manipulación de fechas.
- Se definieron restricciones UNIQUE en **iban** y **pan** para evitar duplicados.
- Se definió la condición NOT NULL en todos los campos para garantizar que no hayan valores vacíos.

```

1  -- Tarea S2.01
2  ----- Nivel 1 -----
3  -- Nivel 1. Ejercicio 1
4  -- Creación de Tabla credit_card
5  CREATE TABLE IF NOT EXISTS credit_card (
6      id VARCHAR(15) PRIMARY KEY, -- Identificador único de la tarjeta
7      nombre VARCHAR(100) NOT NULL, -- Nombre del titular de la tarjeta
8      iban VARCHAR(34) NOT NULL UNIQUE, -- Código IBAN único
9      pan VARCHAR(16) NOT NULL UNIQUE, -- Número PAN de la tarjeta
10     pin VARCHAR(4) NOT NULL, -- PIN de 4 dígitos
11     cvv VARCHAR(4) NOT NULL, -- CVV de 3 o 4 dígitos
12     expiring_date DATE NOT NULL -- Fecha de expiración
13 );

```

Output

#	Time	Action	Message
1	22:04:58	CREATE TABLE IF NOT EXISTS credit_card ( id VARCHAR(15) PRIMARY KEY, -- Identificador único de la tarjeta -- nombre VAR...	0 row(s) affected

- Insertar datos

Al intentar insertar los datos en la tabla `credit_card`, se presentó un error relacionado con el formato de la fecha en la columna `expiring_date`. El formato de fecha proporcionado en el archivo SQL era `MM/DD/YY` (por ejemplo, `10/30/22`), no compatible con el tipo de datos `DATE` en MySQL, que requiere el formato `YYYY-MM-DD`.

```

1  -- Insertamos datos de credit_card
2  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
3  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
4  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
5  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
6  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
7  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
8  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
9  INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
10 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
11 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
12 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
13 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
14 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
15 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
16 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
17 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
18 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (
19 INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES (

```

Output

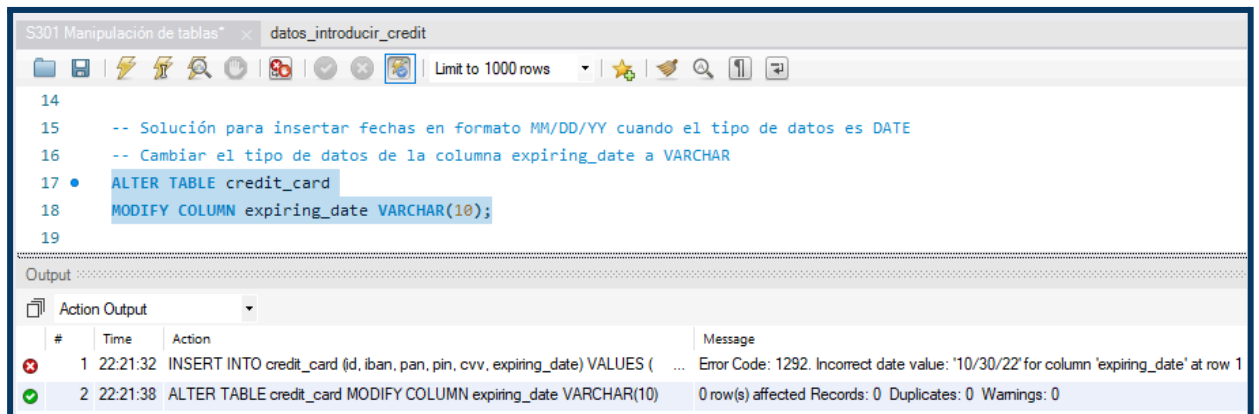
#	Time	Action	Message
1	22:16:16	INSERT INTO credit_card (id, iban, pan, pin, cvv, expiring_date) VALUES ( 'CcU-2938', 'TR301950312213576817638661', '542...	Error Code: 1292. Incorrect date value: '10/30/22' for column 'expiring_date' at row 1

🔧 **Solución aplicada:**

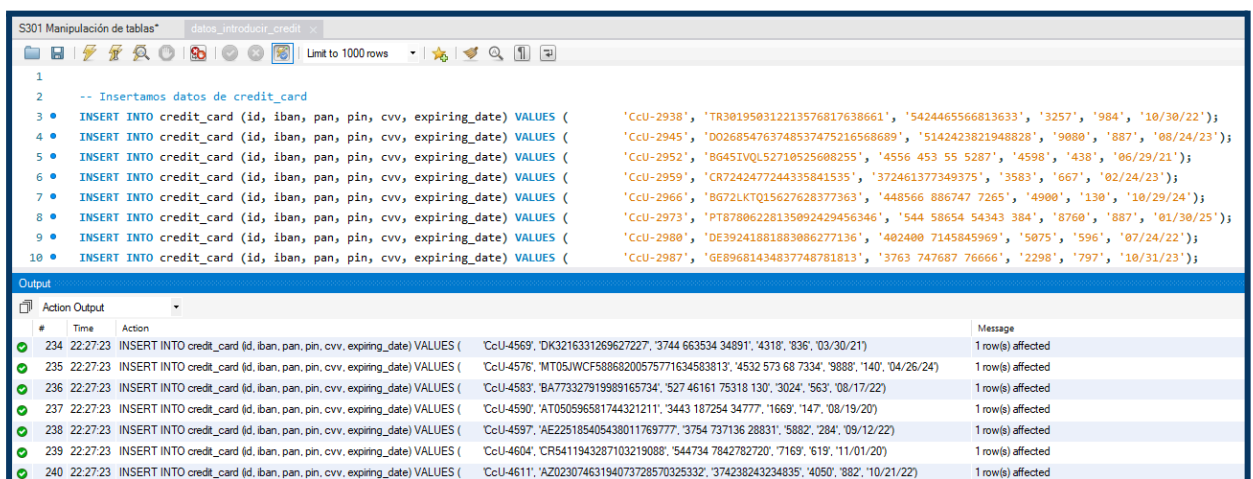
- ➔ Cambiar el tipo de datos de la columna a VARCHAR
- ➔ Insertar los datos
- ➔ Actualizar las fechas al formato correcto (YYYY-MM-DD)
- ➔ Cambiar nuevamente el tipo de datos de la columna a DATE

- Cambiar el tipo de datos de la columna `expiring_date` a `VARCHAR`

```
ALTER TABLE credit_card
MODIFY COLUMN expiring_date VARCHAR(10);
```



- Insertar los datos (código del archivo "datos\_introducir\_credit")



- Actualizar las fechas al formato correcto (YYYY-MM-DD)

→ Por defecto, MySQL Workbench tiene activado el **modo seguro** para evitar modificaciones masivas sin condiciones basadas en claves primarias. Para actualizar los datos, primero es necesario desactivarlo temporalmente

```
SET SQL_SAFE_UPDATES = 0;
```

→ Se actualiza la columna `expiring_date` para modificar el formato de la fecha

```
UPDATE credit_card
SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y');
```

→ Se vuelve a activar el modo seguro

```
SET SQL_SAFE_UPDATES = 1;
```

```

14
15 -- Solución para insertar fechas en formato MM/DD/YY cuando el tipo de datos es DATE
16 -- Cambiar el tipo de datos de la columna expiring_date a VARCHAR
17 • ALTER TABLE credit_card
18   MODIFY COLUMN expiring_date VARCHAR(10);
19
20 -- Modificar el formato de la fecha
21 • SET SQL_SAFE_UPDATES = 0; -- Desactivando el modo seguro que impide actualizar datos sin una condición basada en una clave primaria.
22
23 • UPDATE credit_card
24   SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y');
25
26 • SET SQL_SAFE_UPDATES = 1; -- Activando nuevamente el modo seguro.

```

#	Time	Action	Message
✓ 1	22:29:23	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✓ 2	22:30:07	UPDATE credit_card SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y')	275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0
✓ 3	22:30:11	SET SQL_SAFE_UPDATES = 1	0 row(s) affected

→ Cambiar el tipo de datos nuevamente a **DATE**

```

ALTER TABLE credit_card
MODIFY COLUMN expiring_date DATE;

```

```

14
15 -- Solución para insertar fechas en formato MM/DD/YY cuando el tipo de datos es DATE
16 -- Cambiar el tipo de datos de la columna expiring_date a VARCHAR
17 • ALTER TABLE credit_card
18   MODIFY COLUMN expiring_date VARCHAR(10);
19
20 -- Modificar el formato de la fecha
21 • SET SQL_SAFE_UPDATES = 0; -- Desactivando el modo seguro que impide actualizar datos sin una condición basada en una clave prim
22
23 • UPDATE credit_card
24   SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y');
25
26 • SET SQL_SAFE_UPDATES = 1; -- Activando nuevamente el modo seguro.
27
28 -- Cambiando el tipo de dato nuevamente a Date
29 • ALTER TABLE credit_card
30   MODIFY COLUMN expiring_date DATE;
31

```

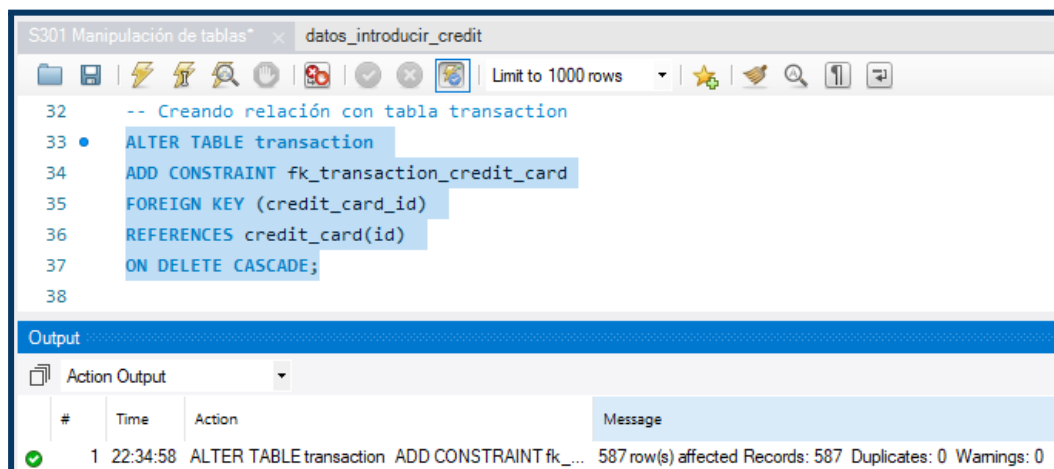
#	Time	Action	Message
✓ 1	22:29:23	SET SQL_SAFE_UPDATES = 0	0 row(s) affected
✓ 2	22:30:07	UPDATE credit_card SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y')	275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0
✓ 3	22:30:11	SET SQL_SAFE_UPDATES = 1	0 row(s) affected
✓ 4	22:32:38	ALTER TABLE credit_card MODIFY COLUMN expiring_date DATE	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

- Creación de las relaciones

La tabla `credit_card` debe tener una relación de **uno a muchos** con la tabla `transaction`, respondiendo a que lógicamente una tarjeta puede estar asociada a múltiples transacciones. Para lograr esto, se debe establecer el campo `credit_card_id` en la tabla `transaction` como clave foránea, vinculándolo con el campo `id` de la tabla `credit_card`.

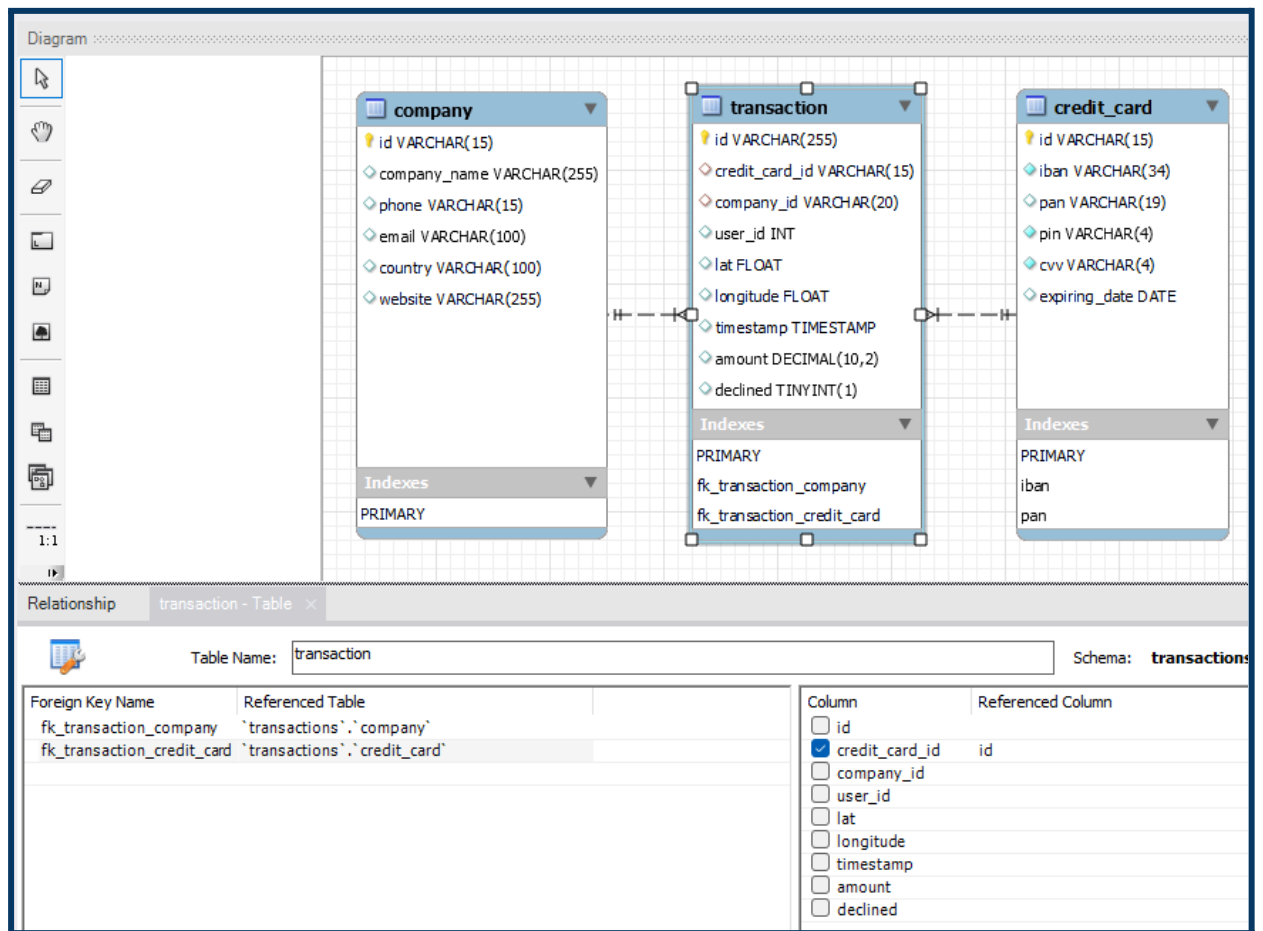
Esta relación no se configuró luego de crear la tabla `credit_card`, debido a que la tabla `transaction` ya contenía datos y resulta necesario que los identificadores de las tarjetas en las transacciones existentes, correspondan a tarjetas registradas. Si se hubiera intentado establecer una relación con registros en `transaction` sin una tarjeta asociada válida en `credit_card`, daría un error para evitar la aparición de registros "huérfanos".

```
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_credit_card
FOREIGN KEY (credit_card_id)
REFERENCES credit_card(id)
ON DELETE CASCADE;
```



Se utiliza `ON DELETE CASCADE` para mantener la integridad referencial y evitar precisamente datos huérfanos.

- Diagrama ER



## Descripción

Este modelo representa una base de datos donde se registran transacciones financieras, vinculadas a tarjetas de crédito y compañías.

- ★ Tabla **credit\_card**
  - Recoge datos relevantes de las tarjetas de crédito
  - Clave **primaria**: **id**
  - Una tarjeta puede estar vinculada a múltiples transacciones (relación 1 a muchos).
- ★ Tabla **company**
  - Recoge datos relevante de las compañías
  - Clave **primaria**: **id**
  - Una compañía puede estar vinculada a múltiples transacciones (relación 1 a muchos).

★ Tabla **transaction**

→ Recoge los datos de las transacciones realizadas.

→ Clave **primaria**: **id**

→ Claves **foráneas**:

◆ **card\_id**: Hace referencia a la tarjeta de crédito utilizada en la transacción, vinculada a la clave primaria de **credit\_card (id)**

◆ **company\_id**: Hace referencia a la compañía que realiza la transacción, vinculada a la clave primaria de **company (id)**

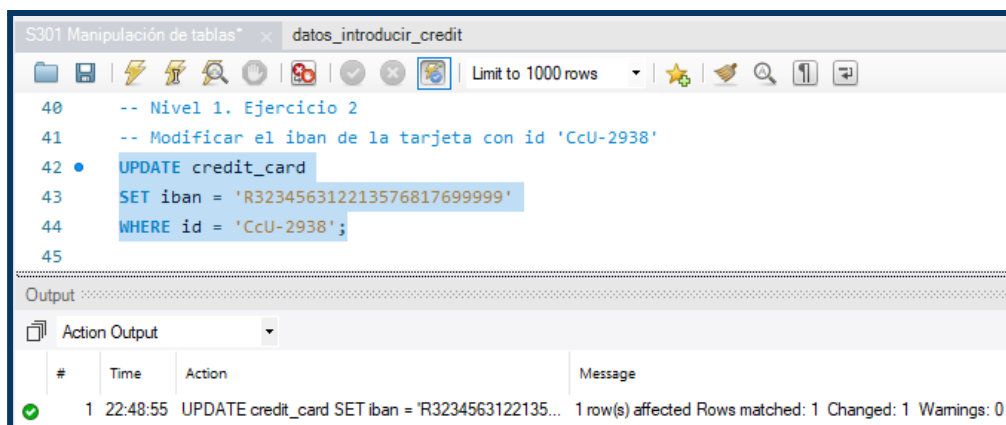
→ Una transacción puede estar vinculada a una única tarjeta y a una única compañía (relación 1 a muchos con ambas).

## 1.2. Ejercicio 2

El departamento de Recursos Humanos ha identificado un error en el número de cuenta del usuario con ID CcU-2938. La información que debe mostrarse para este registro es: R323456312213576817699999. Recuerda mostrar que el cambio se realizó.

- Modificar de datos

```
UPDATE credit_card
SET iban = 'R323456312213576817699999'
WHERE id = 'CcU-2938';
```



- Comprobar cambio realizado

```
SELECT * FROM credit_card
WHERE id = 'CcU-2938';
```



S301 Manipulación de tablas\* x datos\_introducir\_credit

Limit to 1000 rows

```

41 -- Modificar el iban de la tarjeta con id 'CcU-2938'
42 • UPDATE credit_card
43 SET iban = 'R323456312213576817699999'
44 WHERE id = 'CcU-2938';
45
46 -- Comprobar la modificación hecha
47 • SELECT * FROM credit_card
48 WHERE id = 'CcU-2938';
49

```

Result Grid

	id	iban	pan	pin	cvv	expiring_date
▶	CcU-2938	R323456312213576817699999	5424465566813633	3257	984	2022-10-30
*	NULL	NULL	NULL	NULL	NULL	NULL

credit\_card 13 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:48:55	UPDATE credit_card SET iban = 'R323456312...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓ 2	22:50:45	SELECT * FROM credit_card WHERE id = 'CcU...	1 row(s) returned

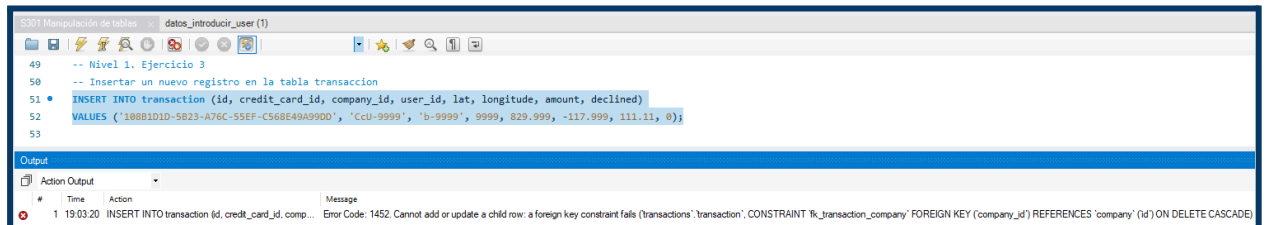
### 1.3. Ejercicio 3

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

Id	108B1D1D-5B23-A76C-55EF-C568E49A99DD
credit_card_id	CcU-9999
company_id	b-9999
user_id	9999
lato	829.999
longitud	-117.999
amunt	111.11
declined	0

- Insertar nuevo registro en transaction

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



Al intentar insertar el registro en la tabla *transaction*, se produce el siguiente error :

**Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('transactions`.`transaction`, CONSTRAINT `fk\_transaction\_company` FOREIGN KEY (`company\_id`) REFERENCES `company` (`id`) ON DELETE CASCADE)**

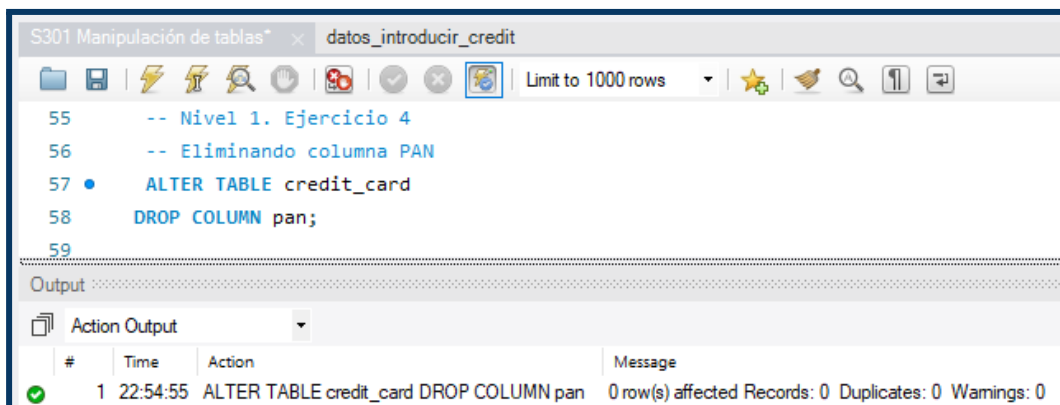
Este error ocurre porque el id 'b-9999' (columna *company\_id*) no existe en la tabla *company* (columna *id*), lo cual no es compatible con la relación establecida entre ambas tablas. Al ser *company\_id* una clave foránea, el valor de esta columna en cada registro debe estar contenido en la tabla *company*; como este valor no existe, no se puede completar la inserción.

#### 1.4. Ejercicio 4

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

- Eliminar columna

```
ALTER TABLE credit_card
DROP COLUMN pan;
```



- Comprobar cambio realizado

DESCRIBE transaction;

S301 Manipulación de tablas\* x datos\_introducir\_credit

Limit to 1000 rows

```

56 -- Eliminando columna PAN
57 • ALTER TABLE credit_card
58 DROP COLUMN pan;
59
60 -- Comprobando que se eliminó la columna
61 • DESCRIBE transaction;
62

```

Result Grid

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(15)	YES	MUL	NULL	
company_id	varchar(20)	YES	MUL	NULL	
user_id	int	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint(1)	YES		NULL	

Result 14 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:54:55	ALTER TABLE credit_card DROP COLUMN pan	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
✓ 2	22:56:50	DESCRIBE transaction	9 row(s) returned

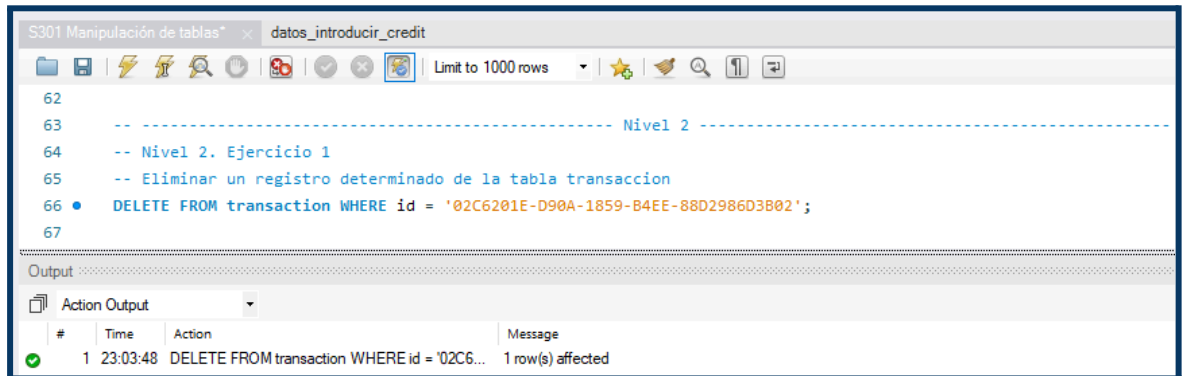
## 2. Nivel 2

### 2.1. Ejercicio 1

Elimina de la tabla transacción el registro con ID  
02C6201E-D90A-1859-B4EE-88D2986D3B02 de la base de datos.

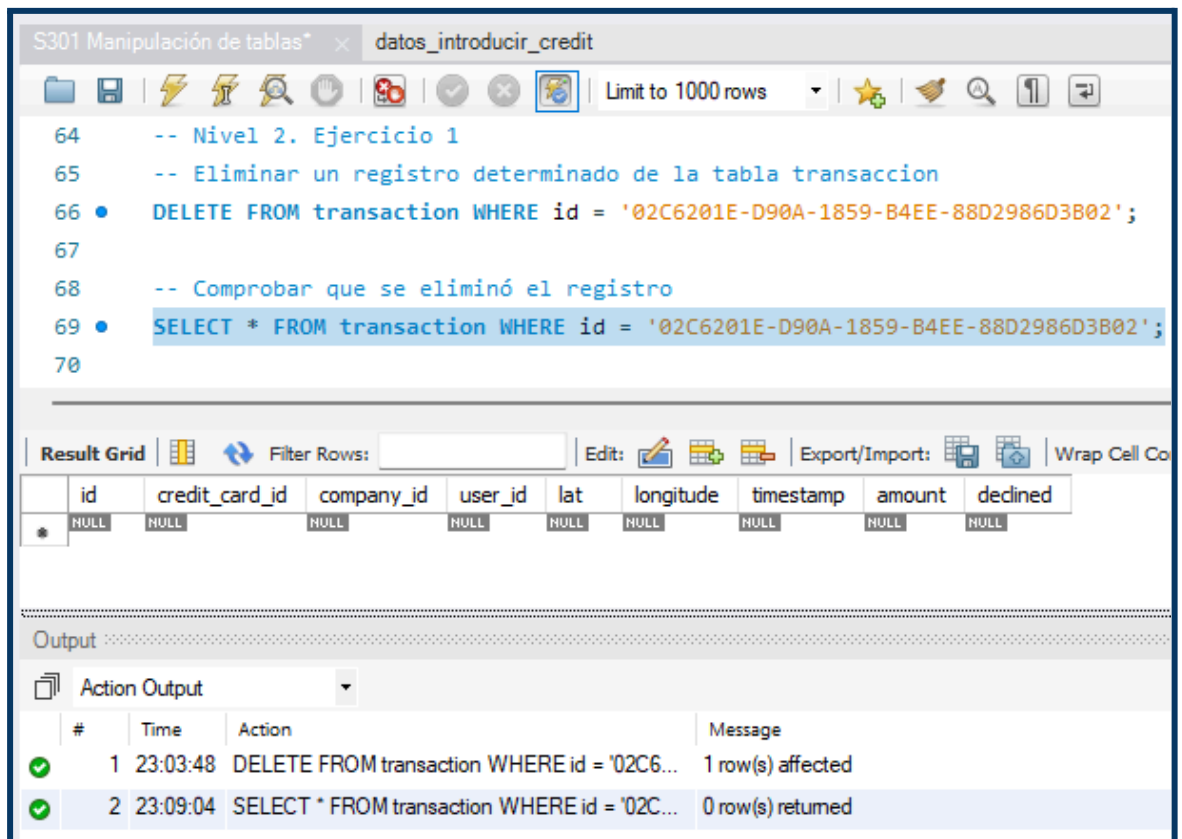
- Eliminar registro

```
DELETE FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
```



- Comprobar cambio realizado

```
SELECT * FROM transaction WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';
```

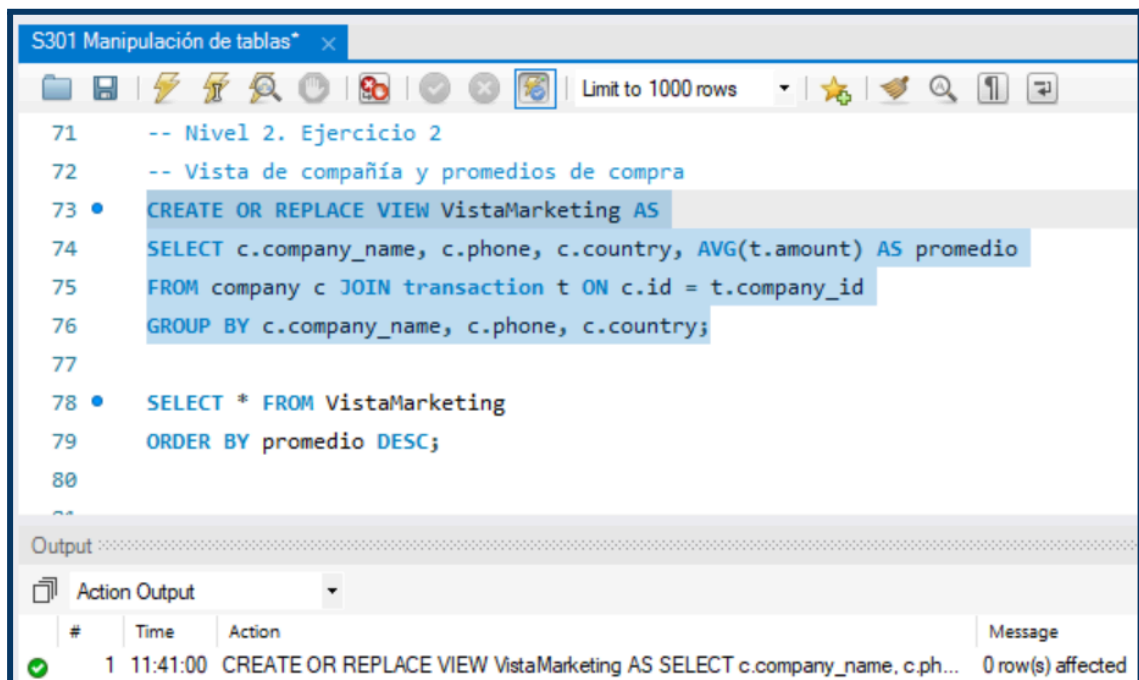


## 2.2. Ejercicio 2

La sección de marketing desea tener acceso a información específica para realizar análisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles clave sobre las compañías y sus transacciones. Será necesaria que crees una vista llamada VistaMarketing que contenga la siguiente información: Nombre de la compañía. Teléfono de contacto. País de residencia. Media de compra realizado por cada compañía. Presenta la vista creada, ordenando los datos de mayor a menor promedio de compra.

- Crear la vista

```
CREATE OR REPLACE VIEW VistaMarketing AS
SELECT c.company_name, c.phone, c.country, AVG(t.amount) AS promedio
FROM company c JOIN transaction t ON c.id = t.company_id
GROUP BY c.company_name, c.phone, c.country;
```



- Mostrar resultados de la vista

```
SELECT * FROM VistaMarketing
ORDER BY promedio DESC;
```

S301 Manipulación de tablas\*

Limit to 1000 rows

```

71 -- Nivel 2. Ejercicio 2
72 -- Vista de compañía y promedios de compra
73 • CREATE OR REPLACE VIEW VistaMarketing AS
74   SELECT c.company_name, c.phone, c.country, AVG(t.amount) AS promedio
75   FROM company c JOIN transaction t ON c.id = t.company_id
76   GROUP BY c.company_name, c.phone, c.country;
77
78 • SELECT * FROM VistaMarketing
79   ORDER BY promedio DESC;
80

```

Result Grid

	company_name	phone	country	promedio
▶	Eget Ipsum Ltd	03 67 44 56 72	United States	473.075000
	Non Magna LLC	06 71 73 13 17	United Kingdom	468.345000
	Sed Id Limited	07 28 18 18 13	United States	461.210000
	Justo Eu Arcu Ltd	08 42 56 71 52	Italy	443.635000
	Eget Tincidunt Dui Institute	05 35 93 32 44	Netherlands	442.520000
	Viverra Donec Foundation	03 33 12 32 73	United Kingdom	442.280000
	Vestibulum Lorem PC	02 02 87 33 40	Belgium	434.060000
	Aliquet Diam Limited	02 76 61 47 46	United States	425.640000
	Maecenas Malesuada Fringilla Inc.	09 38 53 76 61	Netherlands	408.620000
	Non Ante LLP	08 89 47 65 08	Sweden	407.790000
	Egestas Nunc Sed Limited	06 01 02 70 47	Italy	406.110000
	Nunc Sit Incorporated	07 28 42 63 63	Norway	405.355000
	Magna A Neque Industries	04 14 44 64 62	Australia	396.315000
	Amet Luctus Vulputate Foundation	03 18 54 24 19	Canada	390.325000
	Aliquam PC	01 45 73 52 16	Germany	385.265000
	Neque Tellus Incorporated	04 43 18 34 19	Ireland	364.005000
	Placerat LLP	05 43 67 24 41	Netherlands	357.080000
	Elit Etiam Laoreet Associates	07 69 74 17 45	Canada	351.840000
	Fusce Corp.	08 14 97 58 85	United States	350.125000
	Sapien Nunc Pulvinar LLP	08 37 12 58 11	New Zealand	349.655000
	Mauris Institute	05 29 60 36 87	Sweden	346.875000
	Tincidunt Orci Limited	01 78 18 81 44	Norway	326.860000
	Netus Et Malesuada Ltd	02 55 43 68 46	Netherlands	325.335000
	Ditum Eu Corp	03 04 73 67 31	Canada	318.645000

VistaMarketing 6 x

Output

Action Output

#	Time	Action	Message
✓ 1	11:41:00	CREATE OR REPLACE VIEW VistaMarketing AS SELECT c.company_name, c.ph...	0 row(s) affected
✓ 2	11:44:21	SELECT * FROM VistaMarketing ORDER BY promedio DESC LIMIT 0, 1000	100 row(s) returned

### 2.3. Ejercicio 3

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

```
SELECT *  
FROM VistaMarketing  
WHERE country = 'Germany'  
ORDER BY promedio DESC;
```

The screenshot shows a database management tool interface. The top panel displays a SQL query for filtering the 'VistaMarketing' view by country. The middle panel shows the 'Result Grid' with 8 rows of data. The bottom panel shows the 'Output' window with a log of database actions.

**SQL Query:**

```
-- Nivel 2. Ejercicio 3  
-- Filtrar Vista Marketing  
SELECT *  
FROM VistaMarketing  
WHERE country = 'Germany'  
ORDER BY promedio DESC;
```

**Result Grid:**

	company_name	phone	country	promedio
▶	Aliquam PC	01 45 73 52 16	Germany	385.265000
	Ac Industries	09 34 65 40 60	Germany	289.645000
	Rutrum Non Inc.	02 66 31 61 09	Germany	266.900000
	Nunc Interdum Incorporated	05 18 15 48 13	Germany	244.025238
	Augue Foundation	06 88 43 15 63	Germany	240.800000
	Ac Fermentum Incorporated	06 85 56 52 33	Germany	206.465000
	Auctor Mauris Corp.	05 62 87 14 41	Germany	184.310000
	Convallis In Incorporated	06 66 57 29 50	Germany	156.730000

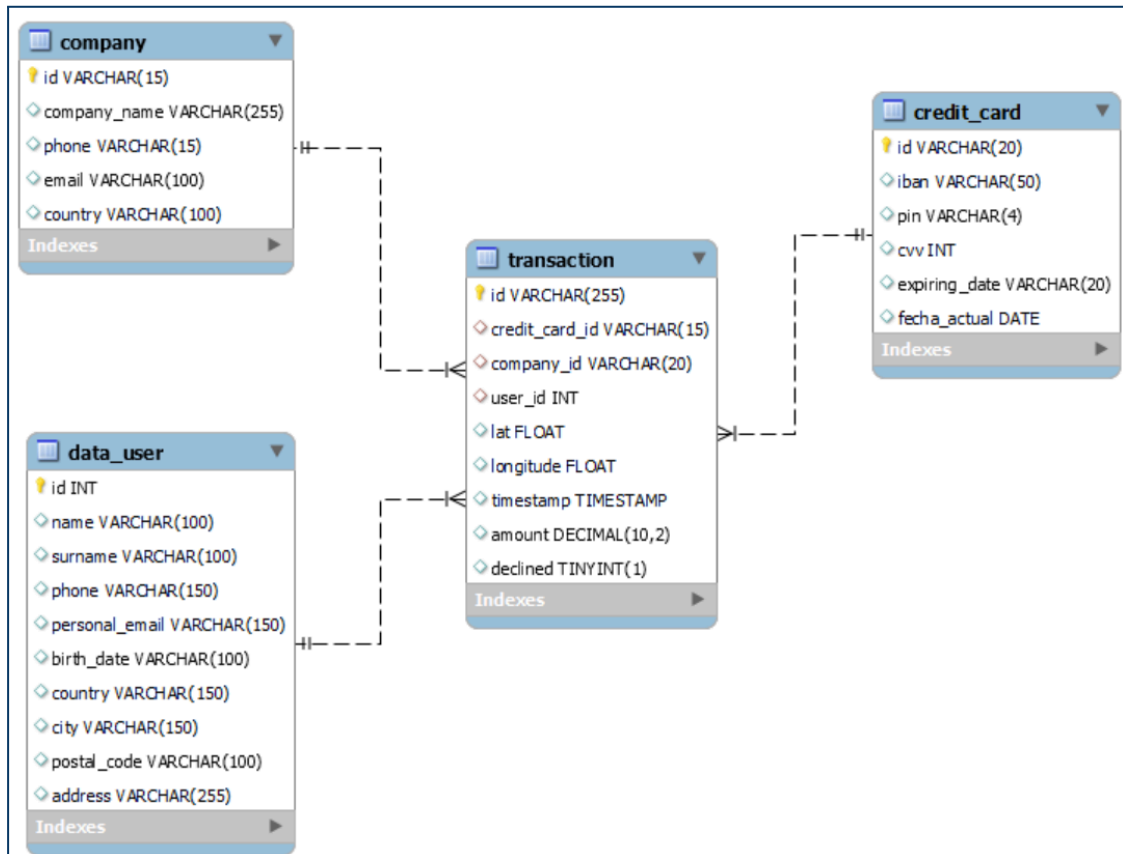
**Output:**

#	Time	Action	Message
✓ 1	11:41:00	CREATE OR REPLACE VIEW VistaMarketing AS SELECT c.company_name, c.ph...	0 row(s) affected
✓ 2	11:44:21	SELECT * FROM VistaMarketing ORDER BY promedio DESC LIMIT 0, 1000	100 row(s) returned
✓ 3	11:48:55	SELECT * FROM VistaMarketing WHERE country = 'Germany' ORDER BY prome...	8 row(s) returned

### 3. Nivel 3

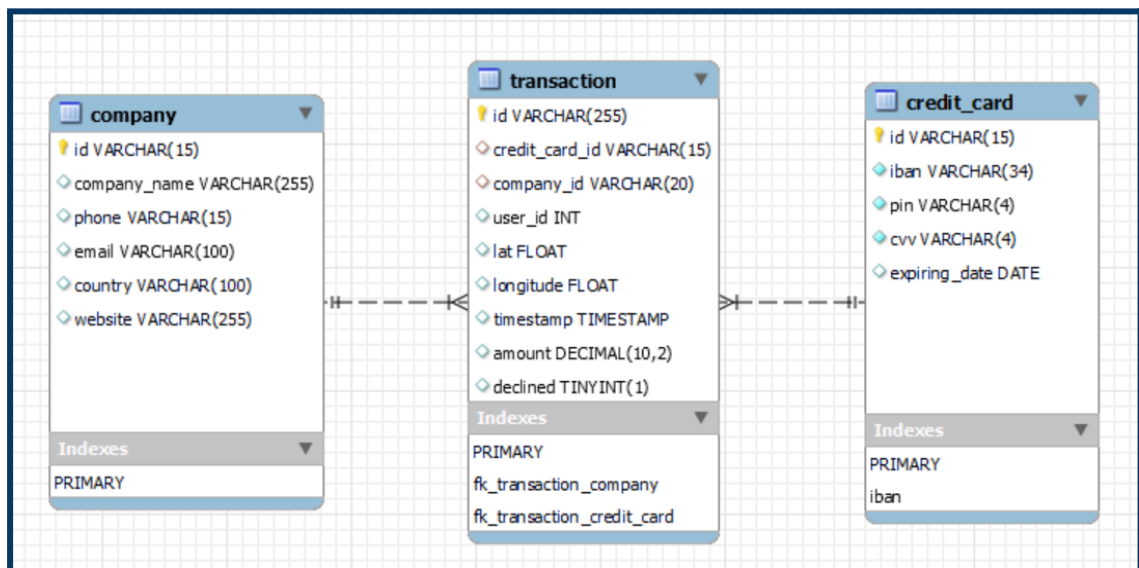
#### 3.1. Ejercicio 1

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:





- Comparación con modelo anterior y definición de modificaciones necesarias por tablas



★ Tabla **user**:

→ Crear tabla con los siguientes atributos

- ◆ id INT PRIMARY KEY
- ◆ name VARCHAR(100)
- ◆ surname VARCHAR(100)
- ◆ phone VARCHAR(150)
- ◆ email VARCHAR(150)
- ◆ birth\_date VARCHAR(100)
- ◆ country VARCHAR(150)
- ◆ city VARCHAR(150)
- ◆ postal\_code VARCHAR(100)
- ◆ address VARCHAR(255)

→ Ingestar data

★ Tabla **transaction**

→ Convertir **user\_id** en **clave foránea**, estableciendo la relación con la tabla **user**.

★ Tabla **company**

→ Eliminar columna **website**

★ Tabla **credit\_card**

→ Modificar la longitud de la columna **id**, de **VARCHAR(15)** a **VARCHAR(20)**

- Modificar la longitud de la columna `iban`, de `VARCHAR(34)` a `VARCHAR(50)`
- Modificar la longitud de la columna `pin`, de `VARCHAR(10)` a `VARCHAR(4)`
- Modificar el tipo de datos de la columna `cvv`, de `VARCHAR(4)` a `INT`
- Modificar el tipo de datos de columna `expiring_date`, de `DATE` a `VARCHAR (20)`
- Modificar de campo obligatorio a campo opcional los atributos `iban`, `pin`, `cvv`
- Agregar columna `fecha_actual` (`DATE`)

- Creación de tabla `user`, ingesta de datos y relación con tabla `transaction`

En el archivo proporcionado para la creación de la tabla `user`

- Se **crea un índice** en la columna `user_id` de la tabla `transaction`

```
CREATE INDEX idx_user_id ON transaction(user_id);
```

- Se crea la tabla `user` con todos los atributos pertinentes y además una **clave foránea** vinculando el campo `id` de `user` con la columna `user_id` de la tabla `transaction`.

```
CREATE TABLE IF NOT EXISTS user (
    id INT PRIMARY KEY,
    ...
    FOREIGN KEY(id) REFERENCES transaction(user_id));
```

Esta instrucción es incorrecta porque la relación entre `user` y `transaction` es de uno a muchos, definiendo que un usuario puede realizar varias transacciones. En este caso, la clave foránea debe estar en `transaction` (no en `user`), para que cada transacción esté correctamente asociada a un único usuario.

A su vez la clave **primaria** de `user` (`id`) no puede ser declarada como clave **foránea** apuntando a `transaction` (`user_id`), ya que cada usuario es único y su identificador no debe depender de la existencia de transacciones.

#### ✚ Para optimizar y corregir la estructura:

- Creación de la tabla `user` sin la clave foránea
  - ◆ Ingestar data
- Adición de la clave foránea en la tabla `transaction`

Al declarar `user_id` como llave foránea, ya no es necesario crear de manera explícita un índice en este campo. El sistema de bases de datos crea automáticamente un índice en los campos que son clave foránea.

- Creación de la tabla `user`

```
CREATE TABLE IF NOT EXISTS user (  
    id INT PRIMARY KEY,  
    name VARCHAR(100),  
    surname VARCHAR(100),  
    phone VARCHAR(150),  
    email VARCHAR(150),  
    birth_date VARCHAR(100),  
    country VARCHAR(150),  
    city VARCHAR(150),  
    postal_code VARCHAR(100),  
    address VARCHAR(255));
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main text area displays a SQL script with line numbers 88 to 102. The script is for creating a table named 'user' with various attributes. The text is highlighted in blue. Below the script, there is an 'Output' section with a dropdown menu set to 'Action Output'. The output table has columns for '#', 'Time', 'Action', and 'Message'. A single row of output is visible, indicating the successful execution of the SQL command.

```
88  ----- Nivel 3 -----  
89  -- Nivel 3. Ejercicio 1  
90  -- Creación de la tabla  
91  CREATE TABLE IF NOT EXISTS user (  
92      id INT PRIMARY KEY,  
93      name VARCHAR(100),  
94      surname VARCHAR(100),  
95      phone VARCHAR(150),  
96      email VARCHAR(150),  
97      birth_date VARCHAR(100),  
98      country VARCHAR(150),  
99      city VARCHAR(150),  
100     postal_code VARCHAR(100),  
101     address VARCHAR(255));  
102
```

#	Time	Action	Message
1	19:46:03	CREATE TABLE IF NOT EXISTS user ( id INT PR...	0 row(s) affected

- Ingesta de datos

Código archivo “datos\_introducir\_user1”

The screenshot shows a database management interface with a tab titled "datos\_introducir\_user (1)". The main area displays a list of SQL INSERT statements for a table named 'user'. Each statement includes fields for id, name, surname, phone, email, birth\_date, country, city, postal\_code, and address. The statements are numbered 4 through 27. Below the SQL editor, the 'Output' section shows the execution results. It includes a table with columns for line number, time, action, and message. The messages indicate that each INSERT statement affected 1 row, and the final statement (line 27) set foreign key checks to 1, affecting 0 rows.

```

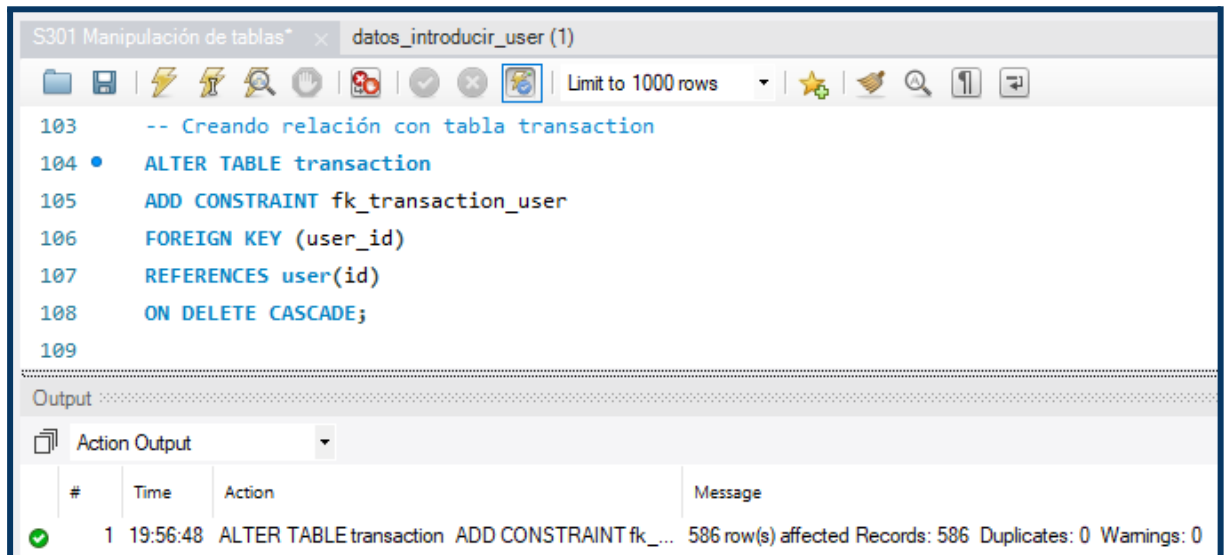
4 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
5 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
6 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
7 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
8 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
9 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
10 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
11 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
12 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
13 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
14 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
15 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
16 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
17 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
18 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
19 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
20 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
21 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
22 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
23 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
24 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
25 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
26 • INSERT INTO user (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
27 • SET foreign_key_checks = 1

```

#	Time	Action	Message
249	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
250	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
251	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
252	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
253	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
254	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
255	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
256	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
257	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
258	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
259	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
260	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
261	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
262	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
263	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
264	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
265	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
266	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
267	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
268	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
269	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
270	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
271	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
272	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
273	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
274	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
275	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
276	19:52:47	INSERT INTO user (id, name, surname, phone, em...	1 row(s) affected
277	19:52:47	SET foreign_key_checks = 1	0 row(s) affected

- Adición de la clave foránea en la tabla `transaction`

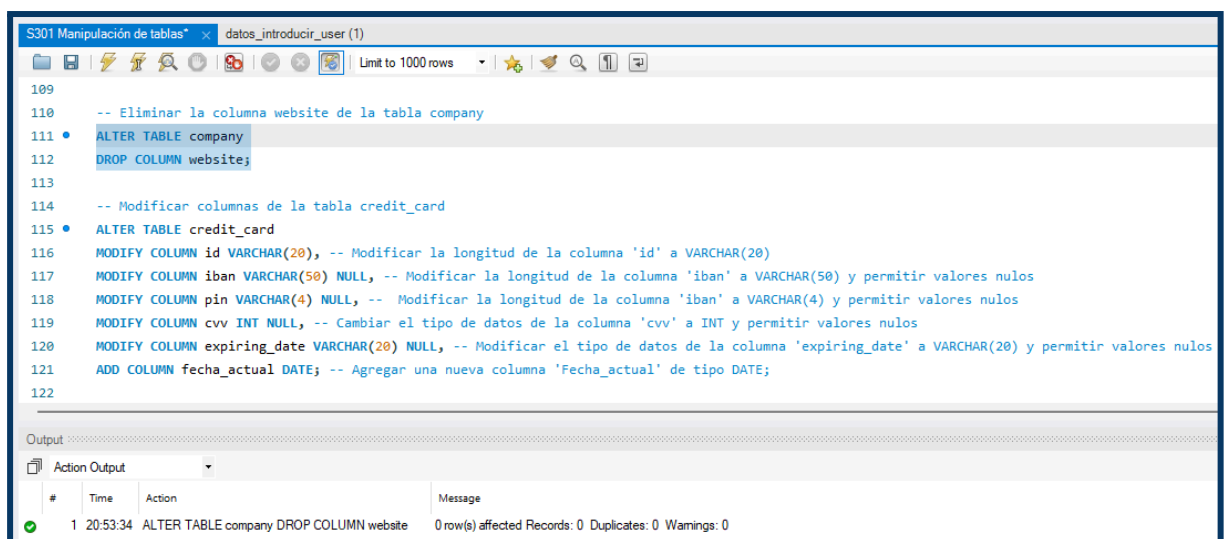
```
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_user
FOREIGN KEY (user_id)
REFERENCES user(id)
ON DELETE CASCADE;
```



- Modificaciones en tablas `company` y `credit_card`

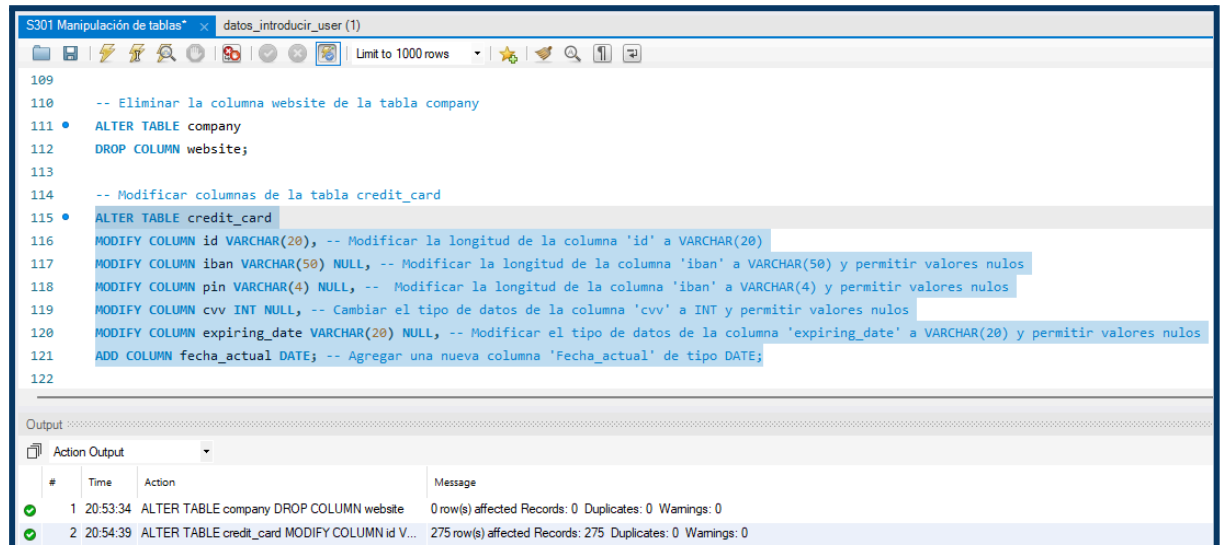
- Eliminar la columna `website` de la tabla `company`

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

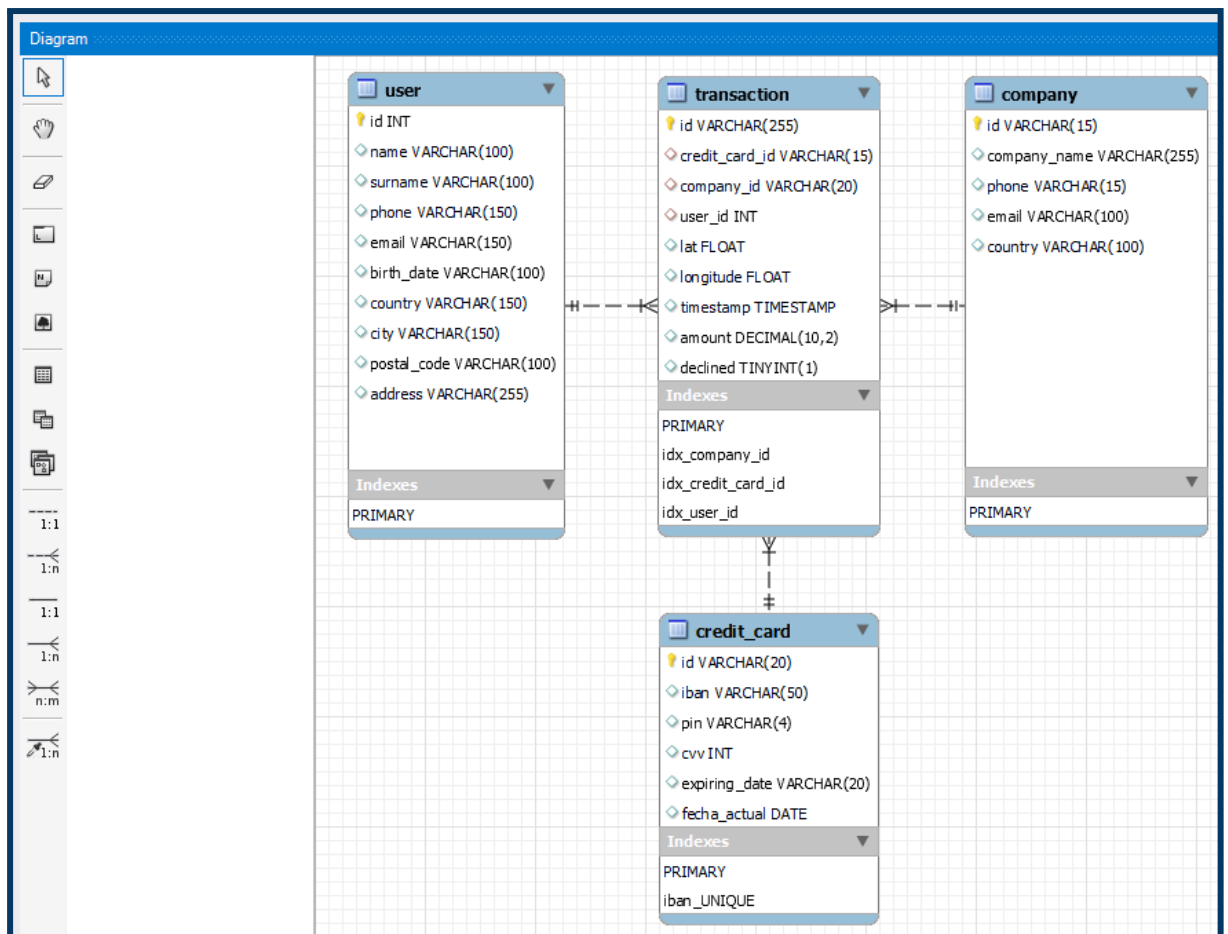


- Modificar columnas de tabla `credit_card`

```
ALTER TABLE credit_card
MODIFY COLUMN id VARCHAR(20),
MODIFY COLUMN iban VARCHAR(50) NULL,
MODIFY COLUMN cvv INT NULL,
MODIFY COLUMN expiring_date VARCHAR(20) NULL,
MODIFY COLUMN pin VARCHAR(4) NULL,
ADD COLUMN fecha_actual DATE;
```



- Generación del Modelo ER



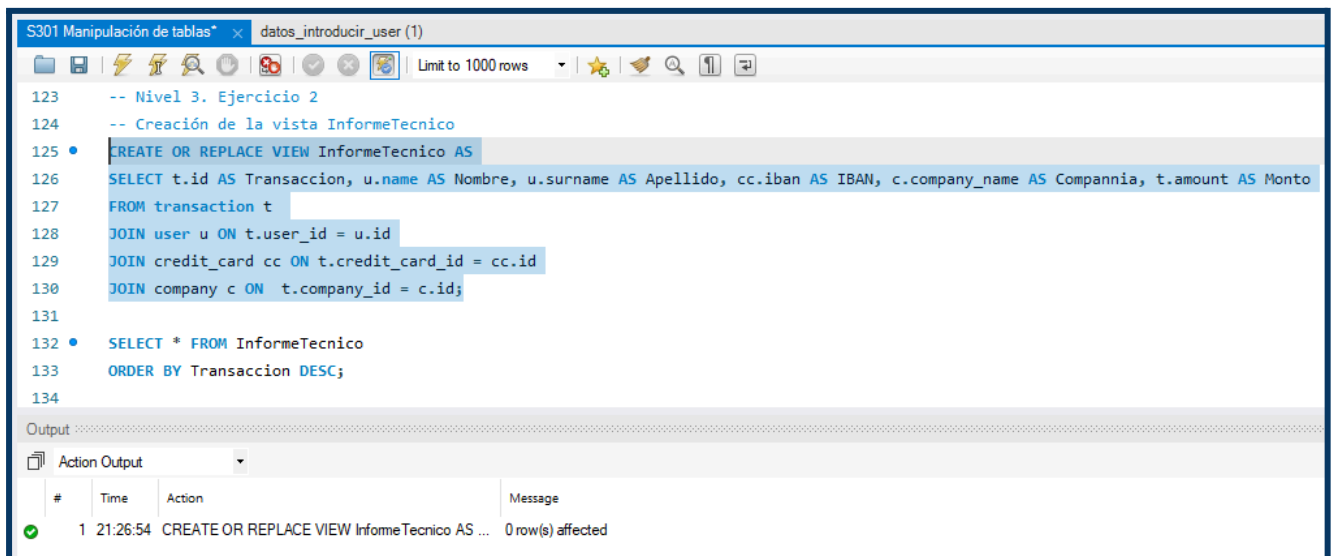
### 3.2. Ejercicio 2

La empresa también te solicita crear una vista llamada "InformeTecnico" que contenga la siguiente información:

- ID de la transacción
- Nombre del usuario/a
- Apellido del usuario/a
- IBAN de la tarjeta de crédito usada.
- Nombre de la compañía de la transacción realizada.
- Asegúrate de incluir información relevante de ambas tablas y utiliza alias para 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.

- Crear la vista

```
CREATE OR REPLACE VIEW InformeTecnico AS
SELECT t.id AS Transaccion, u.name AS Nombre, u.surname AS Apellido,
cc.iban AS IBAN, c.company_name AS Compannia, t.amount AS Monto
FROM transaction t
JOIN user u ON t.user_id = u.id
JOIN credit_card cc ON t.credit_card_id = cc.id
JOIN company c ON t.company_id = c.id;
```



The screenshot shows a SQL IDE window titled "S301 Manipulación de tablas" with a sub-tab "datos\_introducir\_user (1)". The main editor displays SQL code for creating a view and querying it. The code is as follows:

```
123 -- Nivel 3. Ejercicio 2
124 -- Creación de la vista InformeTecnico
125 • CREATE OR REPLACE VIEW InformeTecnico AS
126 SELECT t.id AS Transaccion, u.name AS Nombre, u.surname AS Apellido, cc.iban AS IBAN, c.company_name AS Compannia, t.amount AS Monto
127 FROM transaction t
128 JOIN user u ON t.user_id = u.id
129 JOIN credit_card cc ON t.credit_card_id = cc.id
130 JOIN company c ON t.company_id = c.id;
131
132 • SELECT * FROM InformeTecnico
133 ORDER BY Transaccion DESC;
134
```

Below the code editor is an "Output" section with a dropdown menu set to "Action Output". It contains a table with the following data:

#	Time	Action	Message
✓ 1	21:26:54	CREATE OR REPLACE VIEW InformeTecnico AS ...	0 row(s) affected



- Mostrar resultados de la vista

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

S301 Manipulación de tablas\* x datos\_introducir\_user (1)

Limit to 1000 rows

```

123 -- Nivel 3. Ejercicio 2
124 -- Creación de la vista InformeTecnico
125 • CREATE OR REPLACE VIEW InformeTecnico AS
126 SELECT t.id AS Transaccion, u.name AS Nombre, u.surname AS Apellido, cc.iban AS IBAN, c.company_name AS Compannia, t.amount AS Monto
127 FROM transaction t
128 JOIN user u ON t.user_id = u.id
129 JOIN credit_card cc ON t.credit_card_id = cc.id
130 JOIN company c ON t.company_id = c.id;
131
132 • SELECT * FROM InformeTecnico
133 ORDER BY Transaccion DESC;
```

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

Transaccion	Nombre	Apellido	IBAN	Compannia	Monto
FE96CE47-8D59-381C-4E18-E3CA3D44E8FF	Kenyon	Hartman	DO26854763748537475216568689	Magna A Neque Industries	480.13
FE809ED4-2DB6-55AC-C915-929516E4646B	Molly	Gilliam	SE2813123487163628531121	Nunc Interdum Incorporated	219.83
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	Linus	Willis	KW9485332754781757886242955643	Nunc Interdum Incorporated	42.32
FD89D51B-AE8D-77DC-E450-B8083FBD3187	Hilda	Levy	LT053237077744561475	Malesuada PC	200.72
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	Hedwig	Gilbert	GE84848451582810541526	Neque Tellus Imperdiet Corp.	78.29
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	Hakeem	Hedwig	MD1234119525145401270486	Nunc Interdum Incorporated	335.56
FBD7E0D6-8A6B-F5BC-9C49-EA4B8760100C	Hedwig	Gilbert	MU413233344453432541344788855	Mauris Id Inc.	207.09
FAC76A80-8448-69AA-E892-426C2F12621C	Slade	Poole	MT05JWCF58868200575771634583813	Arcu LLP	304.95
FAAD3FFC-1A17-E141-43D3-359A5BA7CB38	Hedwig	Gilbert	GE90157928843338134463	Lorem Eu Incorporated	149.84
FA053936-75D8-85FA-490D-9B624E1B920A	Hedwig	Gilbert	GT02497653655330848247645975	Non Justo Corp.	151.32
F85A7D75-2778-9D75-D776-3F41A828DE88	Sarah	Beck	VG1468087984174645729577	Ut Semper Foundation	135.93
F843DC08-CCB5-2444-1B4E-5966289FBA8B	Jasper	Landry	VG1468087984174645729577	Ut Semper Foundation	18.08
F5ACD74B-4275-5AA1-2414-6EF417636B98	Nora	Reeves	MD1234119525145401270486	Nunc Interdum Incorporated	148.97
F56FCA4A-0039-9F64-7376-85632B91121B	Lynn	Riddle	CR7242477244335841535	Ut Semper Foundation	294.13
F55B3CE1-3379-E0BF-5AB9-6F4CC2C5479C	Sonya	McKee	EE541536644818872885	Arcu LLP	227.05
F4BCAE41-3B8E-EA8D-9C24-466F7CEB9F9A	Chester	Haynes	CY94263537405015481188625576	Malesuada PC	182.01
F2B3E645-2E6D-E891-9D05-33DBACE58DE4	Heather	Burks	SM6022751049715477062682363	Malesuada PC	267.52
F28E106B-5418-4667-9514-2E2A823ECC65	Hedwig	Gilbert	RO76DAFO6583348580208155	Pede Ultrices Ltd	114.89
F23E386-5A74-63B3-8111-09FC9BA38011	Nero	Mills	HU95215627749276573565556322	Arcu LLP	146.34
F22BB361-E3CD-BC41-DD6A-A4694F175CD8	Aiko	Chaney	MD5723087436783068347555	Non Institute	385.28
F1A598A2-86C5-50A9-F1CE-FB1D69866C39	Craig	Shepherd	R323456312213576817699999	Lorem Eu Incorporated	229.65
EFEECC2E-2A69-AB33-D599-E82AA689E3B3	Alika	Kinney	R323456312213576817699999	Lorem Eu Incorporated	394.59
EF9E08A9-457D-D7C1-41E4-059B31C35CEC	Cassan...	Ferguson	DO68192976973138848171352176	Ut Semper Foundation	274.28
EEAAE515-66B6-B449-5926-54D397988AD9	Ocean	Nelson	AE640696354928782425103	Et Magnis Ltd	316.74
EE8ED5A8-CEC8-9387-524C-A911B8473C1F	Ainsley	Herrera	BH62714428368066765294	Enim Condimentum Ltd	105.51
EE2D8157-D63E-6A91-A363-9CF68B530AD3	Jada	Porter	GI98IMNL122444875373267	Malesuada PC	293.94

InformeTecnico 8 x

Output

Action Output

#	Time	Action	Message
1	21:26:54	CREATE OR REPLACE VIEW InformeTecnico AS...	0 row(s) affected
2	21:27:37	SELECT * FROM InformeTecnico ORDER BY Tra...	586 row(s) returned