

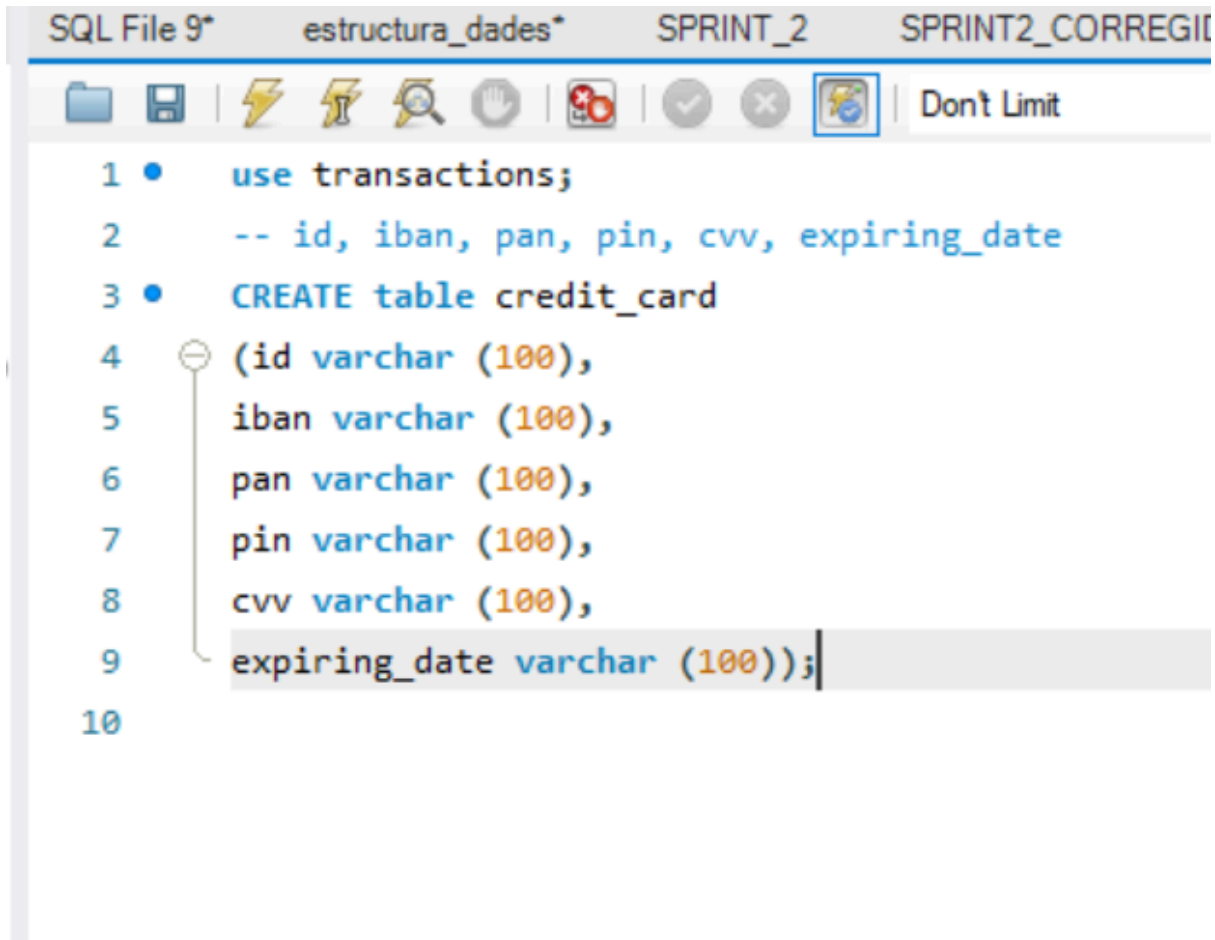
## Sprint 3 Data Analytics

### Nivell 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 tiene que ser capaz de identificar de manera única cada tarjeta y establecer una relación adecuada con las otras dos tablas ("transaction" y "compañía"). 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 de este.

En primer lugar, creamos la tabla en MySQL:

A screenshot of a MySQL IDE window. The title bar shows 'SQL File 9\*', 'estructura\_dades\*', 'SPRINT\_2', and 'SPRINT2\_CORREGI...'. The toolbar includes icons for file operations, execution, and a 'Don't Limit' button. The SQL editor contains the following code:

```
1 • use transactions;
2   -- id, iban, pan, pin, cvv, expiring_date
3 • CREATE table credit_card
4   (id varchar (100),
5    iban varchar (100),
6    pan varchar (100),
7    pin varchar (100),
8    cvv varchar (100),
9    expiring_date varchar (100));
10
```

Los campos de la tabla **credit\_card** son:

**ID:** identificador

**IBAN:** Número Internacional de Cuenta Bancaria.

**PAN:** Número de Cuenta Principal, es el número largo que se encuentra en la tarjeta de crédito o débito y que identifica la cuenta del titular de la tarjeta.

**PIN:** Número de Identificación Personal.

**CVV:** Valor de Verificación de Tarjeta

**Expiring date:** La fecha hasta la cual una tarjeta de crédito o débito es válida.

Utilizamos un tipo de dato **VARCHAR** de **100** caracteres para evitar problemas con números grandes, el máximo es de 255. Luego, ingresamos el documento denominado "datos\_introducir\_credit"

Para verificar que todo esté correcto, hacemos un **SELECT\*** en la tabla, que nos devuelve un resultado de **275 filas**.

```
16 • SELECT *
17 FROM credit_card;
18
```

| id       | iban                         | pan                 | pin  | cvv | expiring_date |
|----------|------------------------------|---------------------|------|-----|---------------|
| CcU-2938 | TR301950312213576817638661   | 5424465566813633    | 3257 | 984 | 10/30/22      |
| CcU-2945 | DO26854763748537475216568689 | 5142423821948828    | 9080 | 887 | 08/24/23      |
| CcU-2952 | BG45IVQL52710525608255       | 4556 453 55 5287    | 4598 | 438 | 06/29/21      |
| CcU-2959 | CR7242477244335841535        | 372461377349375     | 3583 | 667 | 02/24/23      |
| CcU-2966 | BG72LKTQ15627628377363       | 448566 886747 7265  | 4900 | 130 | 10/29/24      |
| CcU-2973 | PT87806228135092429456346    | 544 58654 54343 384 | 8760 | 887 | 01/30/25      |

credit\_card 7 x

Output

Action Output

| # | Time     | Action                    | Message             |
|---|----------|---------------------------|---------------------|
| 1 | 12:31:50 | SELECT * FROM credit_card | 275 row(s) returned |

## Diagrama

En primer lugar, para que haya relación entre las tablas debemos añadir una Primary key a la tabla credit\_card.

```
ALTER table credit_card
ADD PRIMARY KEY(id);
```

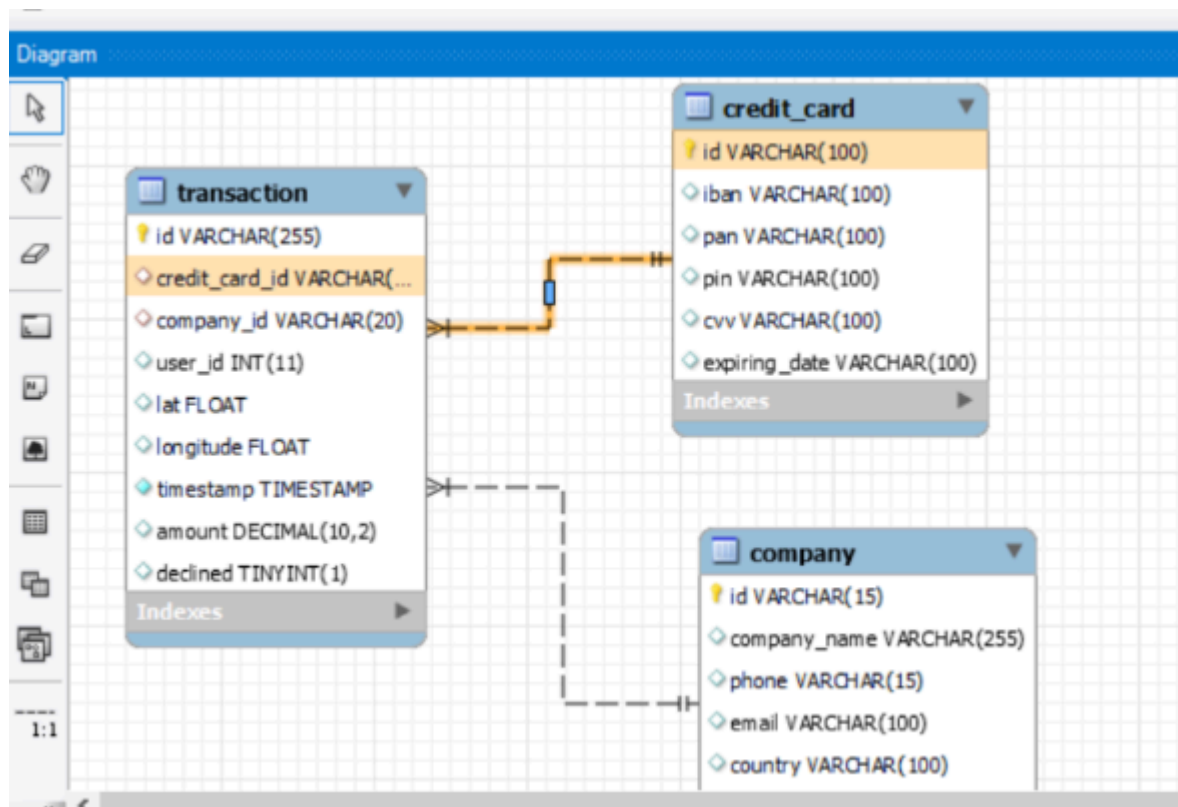
Para ello, alteramos la tabla que necesitamos y añadimos la que será la PK entre paréntesis.

En segundo lugar, utilizamos los siguientes comandos para crear una relación entre la tabla "credit\_card" y "transaction".

```
ALTER TABLE transaction  
ADD CONSTRAINT creditcard  
FOREIGN KEY (credit_card_id)  
REFERENCES credit_card(id);
```

Con Alter Table transaction lo que estamos haciendo es decir que vamos a modificar la estructura de la tabla transaction. Después con Add Constraint estamos añadiendo una nueva restricción a la tabla y la hemos llamado “creditcard”.

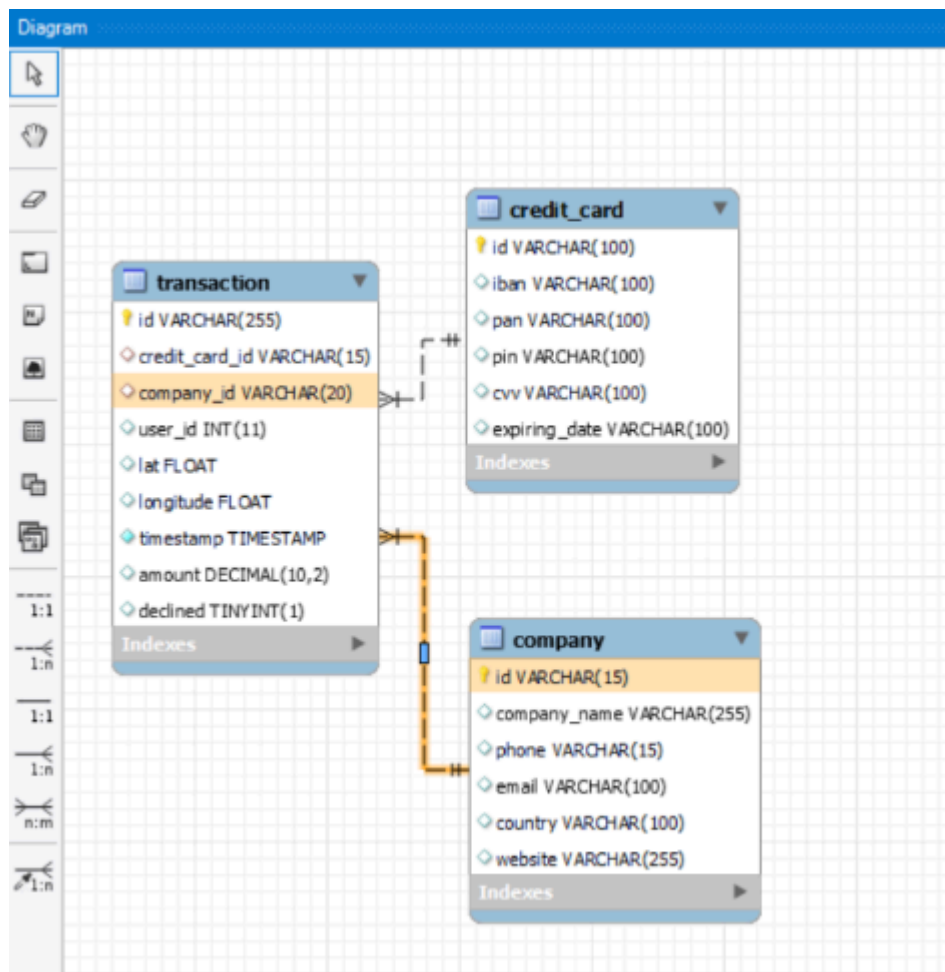
Luego, con Foreign key especificamos que estamos añadiendo clave foránea a la tabla “transaction”. Y que La columna en “transaction” que se está estableciendo como clave foránea es “credit\_card\_id”. Por último, con References credit\_card(id) indicamos que la columna credit\_card\_id en la tabla transaction hace referencia a la columna id en la tabla credit\_card.



Ahora ya podemos observar como la tabla credit\_card se ha unido a la tabla transaction de la PK credit\_card.id a la FK transaction.credit\_card\_id.

Cristina Cumplido Huertas  
Manipulación de tablas

A su vez recordemos que la tabla “transaction” está relacionada con la tabla “company” ¿cómo? pues a través de la PK (id) de la tabla “company” y la Fk (company\_id) de la tabla “transaction”.



La relación desde transacción (para ambas tablas: credit\_card y Company) es de N a 1. Muchas transacciones corresponden a solo 1 credit\_card o solo 1 company.

## -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 tiene que mostrarse para este registro es: R323456312213576817699999. Recuerda mostrar que el cambio se realizó.

En un principio, debemos buscar el registro erróneo:

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

| It Grid                           |                            |                  |      |      |               |
|-----------------------------------|----------------------------|------------------|------|------|---------------|
| Filter Rows: <input type="text"/> |                            |                  |      |      |               |
| Edit:                             |                            |                  |      |      |               |
| Export/Import:                    |                            |                  |      |      |               |
| Wrap Cell Content:                |                            |                  |      |      |               |
| d                                 | iban                       | pan              | pin  | cvv  | expiring_date |
| cU-2938                           | TR301950312213576817638661 | 5424465566813633 | 3257 | 984  | 10/30/22      |
| NULL                              | NULL                       | NULL             | NULL | NULL | NULL          |

| _card 1 x     |          |   |
|---------------|----------|---|
| it            |          |   |
| Action Output |          |   |
| #             | Time     | Action  |
| 1             | 17:59:07 | select* from transactions.credit_card where id = 'CcU-2938' |
|               |          | Message   |
|               |          | 1 row(s) returned   |

Ahora que ya lo hemos ubicado, procedemos a realizar el cambio exigido:

```
5 • UPDATE transactions.credit_card  
6 SET iban = 'R323456312213576817699999'  
7 WHERE id = 'CcU-2938';  
8
```

| input         |          |  |
|---------------|----------|--|
| Action Output |          |  |
| #             | Time     | Action   |
| 1             | 18:01:27 | UPDATE transactions.credit_card SET iban = 'R323456312213576817699999' WHERE id = 'CcU-2938' |
|               |          | Message  |
|               |          | 1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0                                     |

Como podemos observar en la query, el UPDATE es la palabra clave que indica que queremos modificar los datos de la tabla credit\_card.

Con SET indicamos los cambios que queremos realizar, en este caso, cambiar el iban por el número que acaba en -9999. Después, especificamos que este cambio se debe dar en el id 'CcU-2938'.

Finalmente, volvemos a buscar el número de cuenta del usuario para verificar que se han realizado los cambios:

```
39 • SELECT*
40 FROM transactions.credit_card
```

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

|   | id       | iban                      | pan              | pin  | cvv  | expiring_date |
|---|----------|---------------------------|------------------|------|------|---------------|
| ▶ | CcU-2938 | R323456312213576817699999 | 5424465566813633 | 3257 | 984  | 10/30/22      |
| * | NULL     | NULL                      | NULL             | NULL | NULL | NULL          |

credit\_card 2 x

Output

Action Output

| #   | Time     | Action  | Message           |
|-----|----------|---|-------------------|
| ✓ 1 | 18:04:00 | SELECT* FROM transactions.credit_card WHERE id = 'CcU-2938' | 1 row(s) returned |

## -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                                 |
| lat            | 829.999                              |
| longitude      | -117.999                             |
| amount         | 111.11                               |
| declined       | 0                                    |

En primer lugar, debemos ingresar los datos en las tablas "company" y "credit\_card" bajo el comando INSERT INTO y después el valor.

## Cristina Cumplido Huertas

### Manipulación de tablas

```
46 • INSERT INTO transactions.company (id)
47   VALUES ('b-9999');
48
49 • INSERT INTO transactions.credit_card (id)
50   VALUES ('CcU-9999');
51
```

Output

| # | Time     | Action  | Message           |
|---|----------|---|-------------------|
| 1 | 18:15:29 | SELECT* FROM transactions.company WHERE id='b-9999'         | 0 row(s) returned |
| 2 | 18:18:30 | INSERT INTO transactions.company (id) VALUES (b-9999)       | 1 row(s) affected |
| 3 | 18:20:54 | SELECT* FROM transactions.credit_card WHERE id='CcU-9999'   | 0 row(s) returned |
| 4 | 18:21:43 | INSERT INTO transactions.credit_card (id) VALUES (CcU-9999) | 1 row(s) affected |

Después, volvemos a usar el comando INSERT TO para la tabla “transaction”, primero irán los nombres de las columnas y después en VALUES los valores de cada una de ellas. También es importante usar NOW() para obtener la fecha y hora actuales del sistema en el momento en que se ejecuta la consulta. Por otra parte, debemos recordar que si no usamos NOW(), y cargamos los datos, "Timestamp" nos quedaría como NULL.

Finalmente, observamos con el selector de todo en la tabla “transaction” cuyo where id es "108B1D1D-5B23-A76C-55EF-C568E49A99DD" que se han ingresado los datos correctamente.

```
52 -- Ahora ya podemos crear el nuevo usuario en transaction
53 • INSERT INTO transactions.transaction(id,credit_card_id,company_id, user_id, lat, longitude, timestamp, amount, declined)
54   VALUES ('108B1D1D-5B23-A76C-55EF-C568E49A99DD', 'CcU-9999', 'b-9999', '9999', '829.999', '-117.999', now(), '111.11', '0');
55 -- Nos aseguramos de que se ha registrado correctamente
56 • SELECT * FROM transaction
57   WHERE id = "108B1D1D-5B23-A76C-55EF-C568E49A99DD";
58
```

| 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  | 2024-06-18 18:29:54 | 111.11 | 0        |
| * NULL                               | NULL           | NULL       | NULL    | NULL    | NULL      | NULL                | NULL   | NULL     |

transaction 5 x Apply Revert

Output

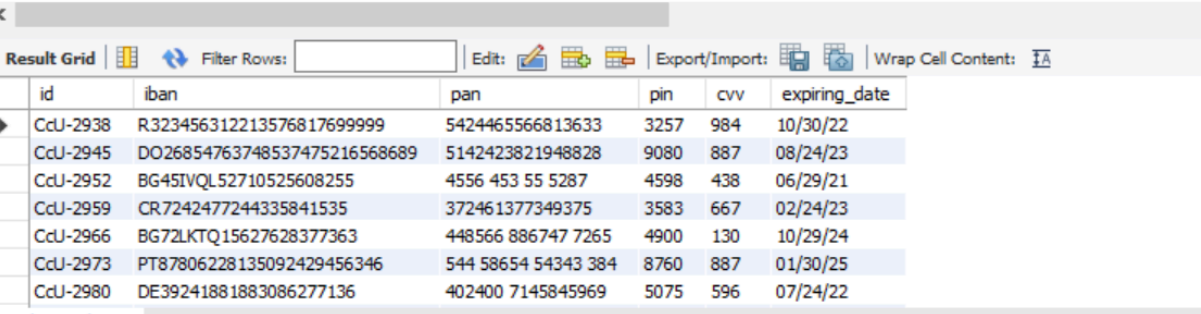
| # | Time     | Action   | Message           |
|---|----------|--|-------------------|
| 1 | 18:15:29 | SELECT* FROM transactions.company WHERE id='b-9999'  | 0 row(s) returned |
| 2 | 18:18:30 | INSERT INTO transactions.company (id) VALUES (b-9999)  | 1 row(s) affected |
| 3 | 18:20:54 | SELECT* FROM transactions.credit_card WHERE id='CcU-9999'  | 0 row(s) returned |
| 4 | 18:21:43 | INSERT INTO transactions.credit_card (id) VALUES (CcU-9999)  | 1 row(s) affected |
| 5 | 18:29:54 | INSERT INTO transactions.transaction(id,credit_card_id,company_id, user_id, lat, longitude, timestamp, amount, declined) | 1 row(s) affected |
| 6 | 18:30:02 | 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.

En primer lugar visualizamos la columna "pan" en la tabla "credit\_card".

```
59 -- Nivel 1. Ejercicio 4. Desde recursos humanos te solicitan eliminar la columna "pan" de la tabl
60 • SELECT *
61 FROM transactions.credit_card;
```



|   | id       | iban                         | pan                 | pin  | cvv | expiring_date |
|---|----------|------------------------------|---------------------|------|-----|---------------|
| ▶ | CcU-2938 | R323456312213576817699999    | 5424465566813633    | 3257 | 984 | 10/30/22      |
|   | CcU-2945 | DO26854763748537475216568689 | 5142423821948828    | 9080 | 887 | 08/24/23      |
|   | CcU-2952 | BG45IVQL52710525608255       | 4556 453 55 5287    | 4598 | 438 | 06/29/21      |
|   | CcU-2959 | CR7242477244335841535        | 372461377349375     | 3583 | 667 | 02/24/23      |
|   | CcU-2966 | BG72LKTQ15627628377363       | 448566 886747 7265  | 4900 | 130 | 10/29/24      |
|   | CcU-2973 | PT87806228135092429456346    | 544 58654 54343 384 | 8760 | 887 | 01/30/25      |
|   | CcU-2980 | DE39241881883086277136       | 402400 7145845969   | 5075 | 596 | 07/24/22      |

credit\_card 6 x


Output

Action Output

| # | Time     | Action                                 | Message             |
|---|----------|--|---------------------|
| 1 | 18:38:03 | SELECT * FROM transactions.credit_card | 276 row(s) returned |

Y ahora, con los comandos ALTER TABLE..DROP COLUMN eliminamos la columna que no nos interesa, en este caso "pan".

```
62
63 • ALTER TABLE credit_card
64 DROP COLUMN pan;
65
```



output

Action Output

| # | Time     | Action                                  | Message  |
|---|----------|---|--|
| 1 | 18:39:26 | ALTER TABLE credit_card DROP COLUMN pan | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 |

Finalmente, nos aseguramos de que se ha llevado a cabo:



## Cristina Cumplido Huertas

### Manipulación de tablas

```
--
66 • SELECT *
67 FROM transactions.credit_card;
```

|   | id       | iban                         | pin  | cvv | expiring_date |
|---|----------|------------------------------|------|-----|---------------|
| ▶ | CcU-2938 | R323456312213576817699999    | 3257 | 984 | 10/30/22      |
|   | CcU-2945 | DO26854763748537475216568689 | 9080 | 887 | 08/24/23      |
|   | CcU-2952 | BG45IVQL52710525608255       | 4598 | 438 | 06/29/21      |
|   | CcU-2959 | CR7242477244335841535        | 3583 | 667 | 02/24/23      |
|   | CcU-2966 | BG72LKTQ15627628377363       | 4900 | 130 | 10/29/24      |
|   | CcU-2973 | PT87806228135092429456346    | 8760 | 887 | 01/30/25      |
|   | CcU-2980 | DE39241881883086277136       | 5075 | 596 | 07/24/22      |

credit\_card 7 x

Output

| # | Time     | Action                                 | Message             |
|---|----------|--|---------------------|
| 1 | 18:41:07 | SELECT * FROM transactions.credit_card | 276 row(s) returned |

## Nivell 2

### - Ejercicio 1

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

En primer lugar, visualizamos el registro a eliminar y después procedemos a ello bajo el comando DELETE:

```
69 -- Nivel 2. Ejercicio 1. Elimina de la tabla transaction el registro con ID 02C6201E-D90A-1859-B4EE-*88D
70 • SELECT *
71 FROM transactions.transaction
72 WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02';
73
74 • DELETE
75 FROM transactions.transaction
76 WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02';
77
```

Output

| # | Time     | Action   | Message           |
|---|----------|--|-------------------|
| 1 | 18:43:53 | SELECT * FROM transactions.transaction WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02' | 1 row(s) returned |
| 2 | 18:45:00 | DELETE FROM transactions.transaction WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02'   | 1 row(s) affected |

Si lo volvemos a buscar ya no aparece.

```
--  
78 • SELECT *  
79 FROM transactions.transaction  
80 WHERE id='02C6201E-D90A-1859-B4EE-88D2986D3B02';  
81
```

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

| id   | credit_card_id | company_id | user_id | lat  | longitude | timestamp | amount | declined |
|------|----------------|------------|---------|------|-----------|-----------|--------|----------|
| NULL | NULL           | NULL       | NULL    | NULL | NULL      | NULL      | NULL   | NULL     |

## - 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 media de compra.

Crear vista:

```
-- Nivel 2. realizar analisis y estrategias efectivas. Se ha solicitado crear una vista que proporcione detalles  
• CREATE VIEW VistaMarketing AS  
  SELECT company.company_name, company.phone, company.country, ROUND(AVG(transaction.amount), 2) AS media  
  FROM transactions.company  
  JOIN transactions.transaction ON company.id = transaction.company_id  
  WHERE transaction.declined = 0  
  GROUP BY company.company_name, company.phone, company.country;
```

Con el comando CREATE vista creamos una vista llamada VistaMarketing, donde hemos seleccionado el nombre de la compañía, teléfono , país y la media de las transacciones. Después, por medio de una JOIN hemos unido la tabla “transaction” y “company”, hemos puesto el la condición de que no haya transacciones rechazadas con el declined = 0, las hemos agrupado por nombre, teléfono y país.

Mostramos vista ordenada por su media de mayor a menor:

## Cristina Cumplido Huertas

### Manipulación de tablas

```
90 • SELECT *
91 FROM vistaMarketing
92 ORDER BY media DESC;
```

Result Grid

|   | company_name              | phone          | country        | media  |
|---|---------------------------|----------------|----------------|--------|
| ▶ | Eget Ipsum Ltd            | 03 67 44 56 72 | United States  | 481.86 |
|   | Sed Id Limited            | 07 28 18 18 13 | United States  | 477.51 |
|   | Neque Tellus Incorporated | 04 43 18 34 19 | Ireland        | 477.10 |
|   | Nunc Sit Incorporated     | 07 28 42 63 63 | Norway         | 461.83 |
|   | Non Magna LLC             | 06 71 73 13 17 | United Kingdom | 458.74 |

vistaMarketing 13 x

Output

Action Output

| #   | Time     | Action   | Message             |
|-----|----------|--|---------------------|
| ✓ 1 | 22:55:29 | SELECT * FROM vistaMarketing ORDER BY media DESC | 101 row(s) returned |

## - Ejercicio 3

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

En esta ocasión, mostramos la selección de vistaMarketing de las compañías, cuyo país sea "Germany". Y nos da un resultado de 8.

```
5 • SELECT *
6 FROM transactions.vistaMarketing
7 WHERE country = 'Germany';
8
```

Result Grid

|  | company_name               | phone          | country | media  |
|--|----------------------------|----------------|---------|--------|
|  | Ac Industries              | 09 34 65 40 60 | Germany | 396.15 |
|  | Auctor Mauris Corp.        | 05 62 87 14 41 | Germany | 308.99 |
|  | Ac Fermentum Incorporated  | 06 85 56 52 33 | Germany | 293.57 |
|  | Rutrum Non Inc.            | 02 66 31 61 09 | Germany | 266.90 |
|  | Nunc Interdum Incorporated | 05 18 15 48 13 | Germany | 242.95 |
|  | Convallis In Incorporated  | 06 66 57 29 50 | Germany | 60.99  |
|  | Augue Foundation           | 06 88 43 15 63 | Germany | 15.05  |

vistaMarketing 12 x

Output

Action Output

| # | Time     | Action  | Message           |
|---|----------|---|-------------------|
| 1 | 19:00:53 | SELECT * FROM transactions.vistaMarketing WHERE country = 'Germany' | 8 row(s) returned |

Cristina Cumplido Huertas  
Manipulación de tablas