

Sprint.4 NIVEL 1

Carla Cociña

Ejercicio 1

Descarga los archivos CSV, estúdialos y diseña una base de datos con esquema en estrella que contenga al menos 4 tablas desde las que puedas realizar las siguientes consultas:

Revise los archivos y sus datos, generé un nuevo database con el nombre de **sprint4**

- `CREATE DATABASE sprint4;`
- `USE sprint4;`

Hice un esquema estrella en donde user, credit_card y companies son las tablas de dimensiones la tabla transaction es la tabla de hechos las relaciones son 1:N

Tabla de dimensiones

```
• CREATE TABLE user (  
  id VARCHAR(100) 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));
```

Tabla de dimensiones

```
• CREATE TABLE companies (  
  company_id VARCHAR(100) PRIMARY KEY,  
  company_name VARCHAR(255),  
  phone VARCHAR(15),  
  email VARCHAR(100),  
  country VARCHAR(100),  
  website VARCHAR(255)  
);
```

Tabla de dimensiones

```
• CREATE TABLE credit_card (  
  id VARCHAR(100) PRIMARY KEY,  
  user_id VARCHAR(100),  
  iban VARCHAR(50),  
  pan VARCHAR(50),  
  pin VARCHAR(4),  
  cvv INT,  
  track1 VARCHAR(100),  
  track2 VARCHAR(100),  
  expiring_date VARCHAR(20)  
);
```

Al final la tabla de hechos

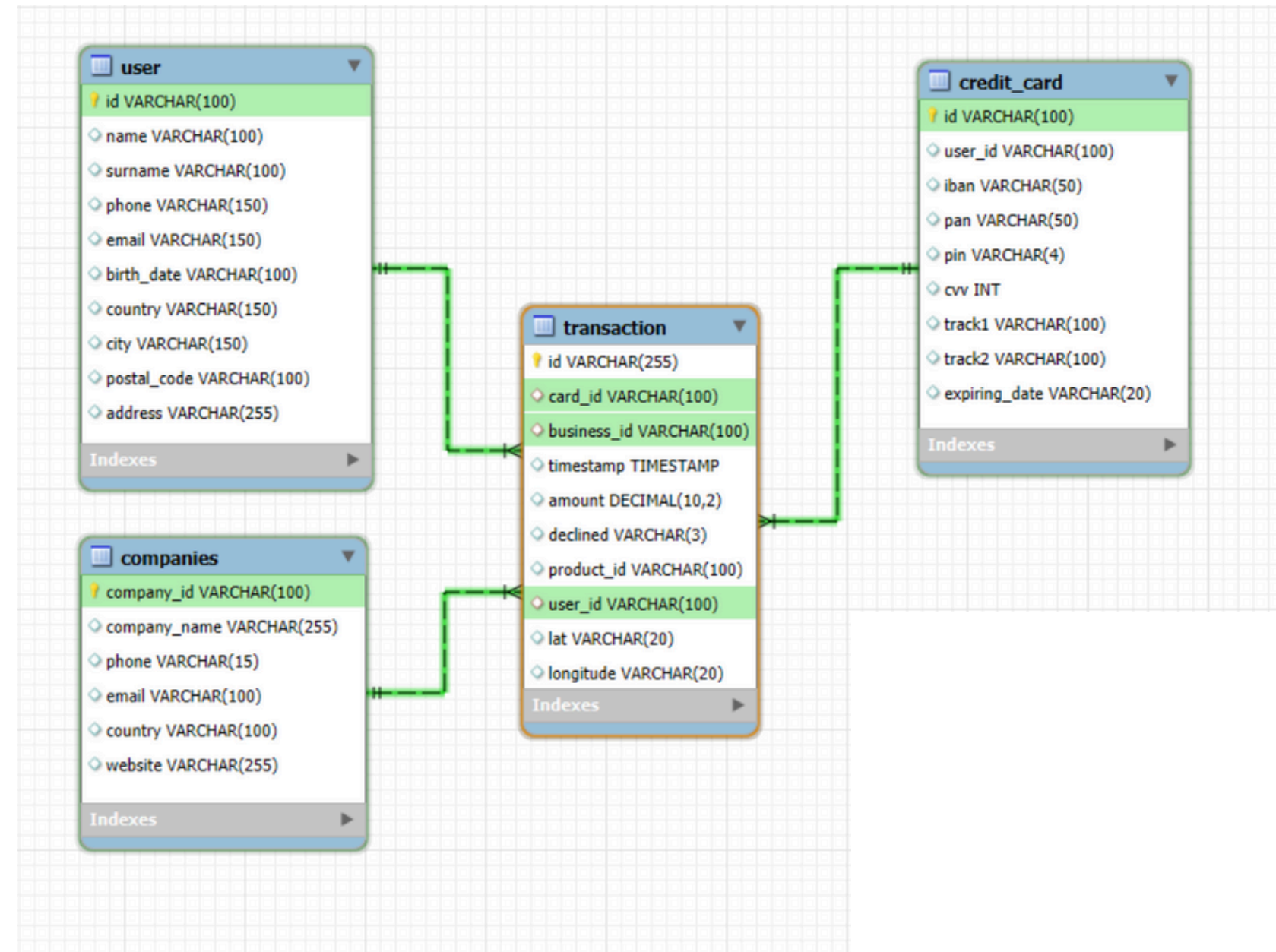
```
• CREATE TABLE transaction (  
  id VARCHAR(255) PRIMARY KEY,  
  card_id VARCHAR(100),  
  business_id VARCHAR(100),  
  timestamp TIMESTAMP,  
  amount DECIMAL(10,2),  
  declined VARCHAR(3),  
  product_id VARCHAR(100),  
  user_id VARCHAR(100),  
  lat VARCHAR(20),  
  longitude VARCHAR(20)  
);
```

Genero las FK para obtener la relacion 1:N

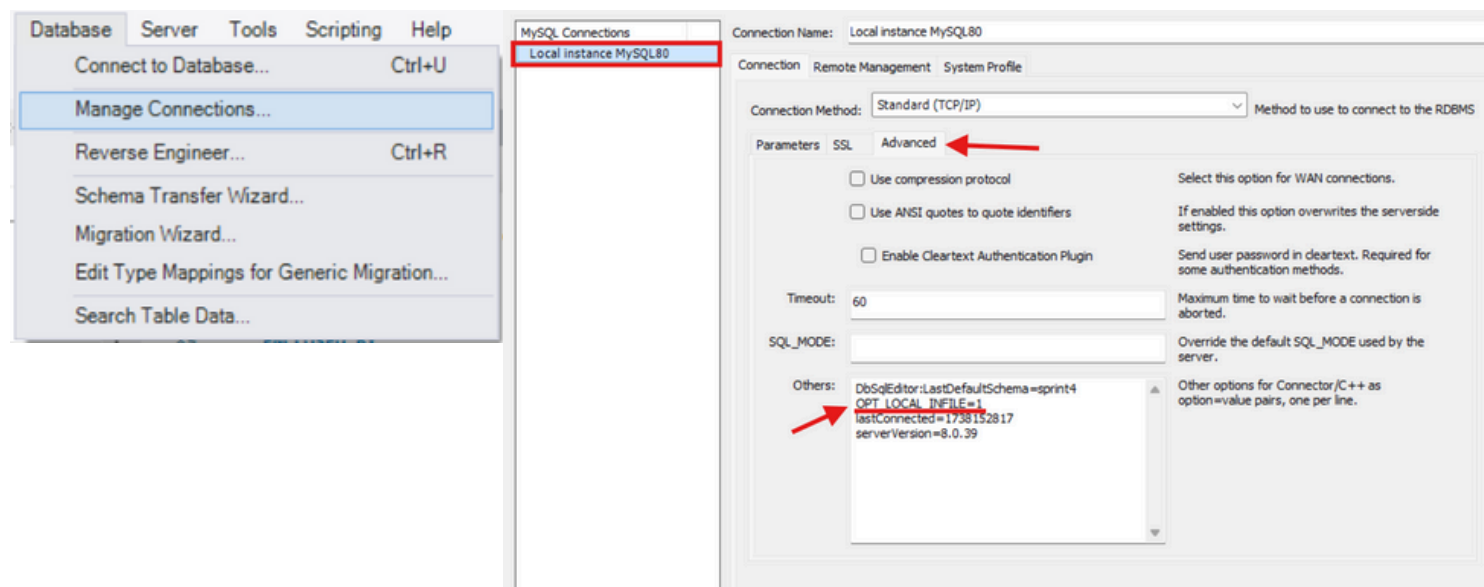
**ALTER TABLE transaction add foreign key (card_id)
references credit_card(id);**

**ALTER TABLE transaction add foreign key (business_id)
references companies(company_id);**

**ALTER TABLE transaction add foreign key (user_id)
references user(id);**



Para poder importar la data desde un archivo local a las tablas hay que desbloquear las restricciones



- -- Con ayuda del compañero Diego logró desbloquear las restricciones al subir datos
`SET GLOBAL local_infile = 1;`
- `LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/users_ca.csv"`
`INTO TABLE user`
`FIELDS TERMINATED BY ','`
`ENCLOSED BY '"'`
`LINES TERMINATED BY '\r\n'`
`IGNORE 1 LINES;`
`-- 75 rows cargadas`
- `LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/users_uk.csv"`
`INTO TABLE user`
`FIELDS TERMINATED BY ','`
`ENCLOSED BY '"'`
`LINES TERMINATED BY '\r\n'`
`IGNORE 1 LINES;`
`-- 50 rows cargadas`
- `LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/users_usa.csv"`
`INTO TABLE user`
`FIELDS TERMINATED BY ','`
`ENCLOSED BY '"'`
`LINES TERMINATED BY '\r\n'`
`IGNORE 1 LINES;`
`-- 150 rows cargadas`

En la tabla **user** cargué los tres scripts **users_ca, uk y usa** ignorando la primera fila porque era la descripción de los nombres de las columnas.

Fields terminated by ',' identifica como se separan los datos.

Enclosed by '"' para que identifique los valores donde parten y terminan

Lines terminated by '\r\n' sirve para que mysql sepa donde termina una fila

Cargue las tablas **credit_card**, **product** y **companies** de la misma manera pero con la terminal **'/n'**.

Con **transaction** estaba separado por **' ; '** fue la única diferencia con las otras tablas

```
• LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/transactions.csv"
  INTO TABLE transaction
  FIELDS TERMINATED BY ';' ←
  ENCLOSED BY ''
  LINES TERMINATED BY '\r\n'
  IGNORE 1 LINES;
```

- ```
LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/credit_cards.csv"
 INTO TABLE credit_card
 FIELDS TERMINATED BY ','
 ENCLOSED BY ''
 LINES TERMINATED BY '\n'
 IGNORE 1 LINES;
```
- ```
LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/products.csv"
  INTO TABLE product
  FIELDS TERMINATED BY ','
  ENCLOSED BY ''
  LINES TERMINATED BY '\n'
  IGNORE 1 LINES;
```
- ```
LOAD DATA LOCAL INFILE "C:/Users/Carla/Curso_reskilling_Data/Sprint-4/companies.csv"
 INTO TABLE companies
 FIELDS TERMINATED BY ','
 ENCLOSED BY ''
 LINES TERMINATED BY '\n'
 IGNORE 1 LINES;
```

# Ejercicio 1

Realizar una subconsulta que muestre todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.

```
155 ● SELECT u.name, u.surname
156 FROM user u LEFT JOIN (SELECT t.user_id, COUNT(t.user_id) as recuento
157 FROM transaction t
158 GROUP BY 1) as t ON u.id = t.user_id
159 WHERE recuento > 30
160 ORDER BY recuento desc;
```





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

|   | name   | surname |
|---|--------|---------|
| ▶ | Hedwig | Gilbert |
|   | Ocean  | Nelson  |
|   | Kenyon | Hartman |
|   | Lynn   | Riddle  |

## Ejercicio 2

Muestra el importe promedio por IBAN de las tarjetas de crédito de la empresa Donec Ltd, utiliza al menos 2 tablas.

```
169 ● SELECT cc.iban, c.company_name, round(avg(t.amount),2) media_iban
170 FROM credit_card cc
171 LEFT JOIN transaction t ON cc.id = t.card_id
172 LEFT JOIN companies c ON t.business_id = c.company_id
173 WHERE c.company_name = 'Donec Ltd'
174 GROUP BY 1;
```

| Result Grid                                                                                                                                                                                                                                                                                                                                                                                                              |                           |              |            |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|--------------|------------|
|   Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:  |                           |              |            |
|                                                                                                                                                                                                                                                                                                                                                                                                                          | iban                      | company_name | media_iban |
| ▶                                                                                                                                                                                                                                                                                                                                                                                                                        | PT87806228135092429456346 | Donec Ltd    | 203.72     |



# Sprint.4 NIVEL 2

*Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas y genera la siguiente consulta:*

Para generar la tabla primero necesito hacer un par de consultas:

- Utilizar un case para generar una condicional que me diga cuando la tarjeta está activa o inactiva.
- Aplicar una función nueva en este curso las “funcionas ventana” para poder dividir estos resultados por categoría y por las ultimas tres transacciones.
- Al crear esta ventana podemos filtrar sobre ella desde where y generar un top 3 (ultimas tres transacciones)

```
/*SELECT t.card_id,
 CASE -- crear una tabla temporal condicional para ver cuantas son activas y cuantas no
 WHEN SUM(t.declined)=3 THEN 'inactiva'
 ELSE 'Activa' -- si no se cumple la primera condicion entonces si son tarjetas activas
 END AS card_status
FROM transaction t
group by 1;

-- Necesitamos saber las ultimas 3 transacciones para eso usamos una funcion ventana
-- Gracias a la compañera Maria que me explico como agregar las funciones ventana
SELECT card_id, declined, timestamp,
 ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_num
FROM transaction
WHERE declined <=3;*/
```

Creo la tabla a partir de credit\_card con la consulta que hice anteriormente y la funcion de ventana la posiciono dentro de una subquery desde from con el alias de **tt**.

Devuelve 275 rows

Después de activar la tabla agrego la **PK** y **FK** para unirla al esquema estrella.

- ALTER TABLE credit\_card\_status ADD PRIMARY KEY (card\_id);
- ALTER TABLE credit\_card\_status ADD FOREIGN KEY (card\_id) references credit\_card(id);

```
22 • CREATE TABLE credit_card_status AS -- crear la tabla nueva desde una query
23 SELECT tt.card_id,
24 CASE -- crear una tabla temporal
25 WHEN SUM(tt.declined)=3 THEN 'inactiva'
26 ELSE 'Activa' -- si no se cumple la primera condicion entonces si son tarjetas activas
27 END AS card_status
28 FROM (SELECT card_id, declined,
29 ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_num
30 FROM transaction) AS tt
31
32 WHERE tt.row_num <=3 -- filtramos la ventana ya que solo queremos 3 o menos
33 GROUP BY tt.card_id;
```

Result Grid

|   | card_id  | card_status |
|---|----------|-------------|
| ▶ | CcU-2938 | Activa      |
|   | CcU-2945 | Activa      |
|   | CcU-2952 | Activa      |
|   | CcU-2959 | Activa      |
|   | CcU-2966 | Activa      |

credit card status 7 x

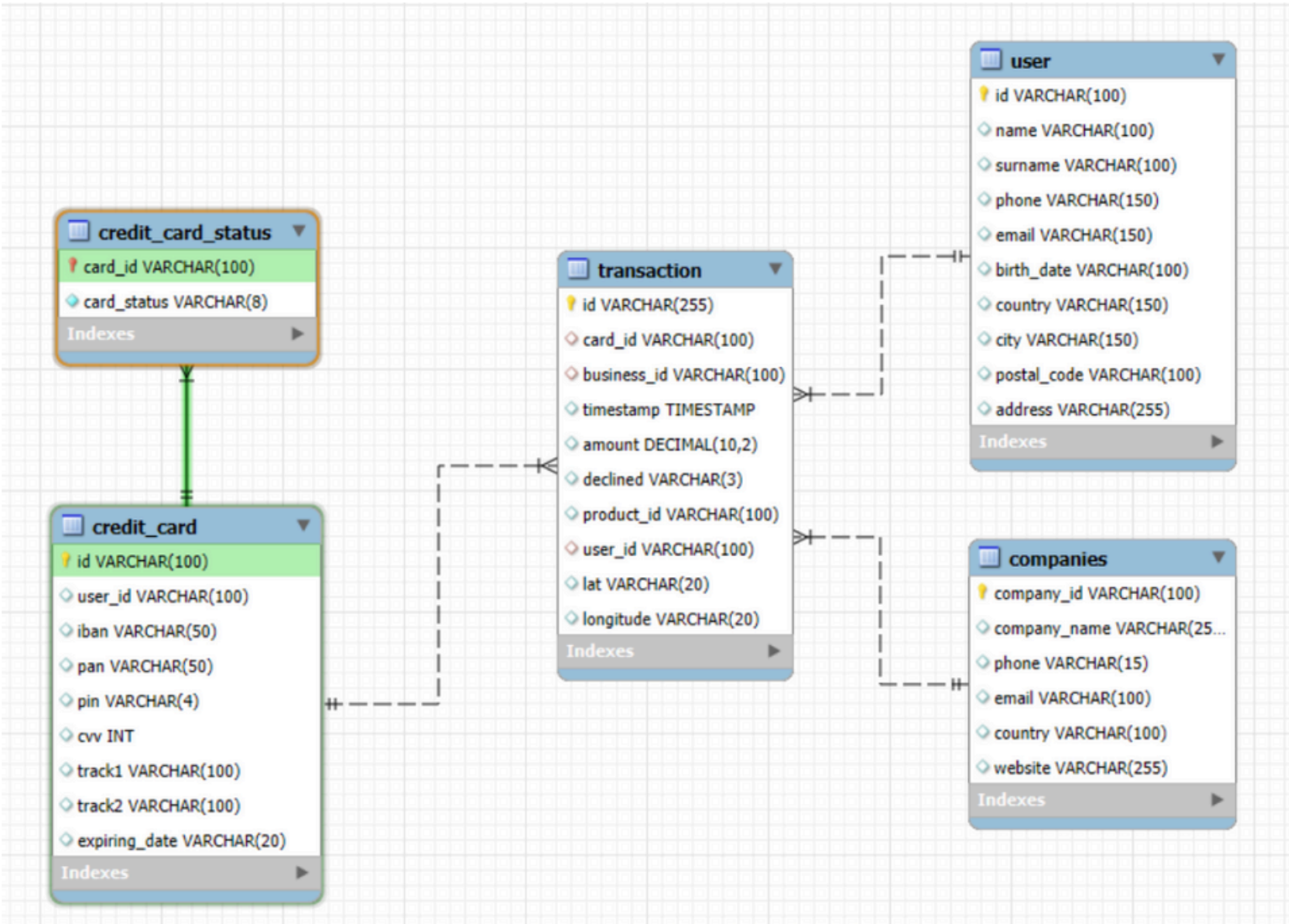
Output

Action Output

| #    | Time     | Action                                                        | Message             |
|------|----------|---------------------------------------------------------------|---------------------|
| ✓ 31 | 12:04:15 | SELECT * FROM credit_card_status WHERE card_status = 'activa' | 275 row(s) returned |



# ESQUEMA FINAL NIVEL 2



# Ejercicio 1

¿Cuántas tarjetas están activas?

```
45 • SELECT COUNT(card_status)
46 FROM credit_card_status
47 WHERE card_status = 'activa';
```

Result Grid

|   | COUNT(card_status) |
|---|--------------------|
| ▶ | 275                |

Result 10 x

Output

| #    | Time     | Action                                                                          | Message           |
|------|----------|---------------------------------------------------------------------------------|-------------------|
| ✓ 35 | 10:34:02 | SELECT COUNT(card_status) FROM credit_card_status                               | 1 row(s) returned |
| ✓ 36 | 10:36:32 | SELECT COUNT(card_status) FROM credit_card_status WHERE card_status = 'activas' | 1 row(s) returned |
| ✓ 37 | 10:36:41 | SELECT COUNT(card_status) FROM credit_card_status WHERE card_status = 'activa'  | 1 row(s) returned |

Desde la creación de la tabla se puede ver que todas las tarjetas están activas ya que ninguna cumple la condición que hice en el CASE anterior, pero si fuera una base de datos con miles de rows lo mejor sería hacer una consulta desde where filtrando por las tarjetas **activas**.

En este caso el resultado es 275 tarjetas activas