MINU CAMPOY
11/2025

SPRINT 4

NIVEL 1

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

Crear database y tablas:

```sql
-- Nivel 1
CREATE DATABASE sprint_4;

-- Crear tablas + upload data:

-- TRANSACTION / id; card_id; business_id; timestamp; amount; declined; product_ids; user_id; lat; longitude
    CREATE TABLE IF NOT EXISTS transaction (
        id VARCHAR(255) PRIMARY KEY,
        card_id VARCHAR(15) REFERENCES credit_card(id),
        business_id VARCHAR(20),
        timestamp TIMESTAMP,
        amount DECIMAL(10, 2),
        declined BOOLEAN,
        product_id VARCHAR(15) REFERENCES products,
        user_id INT REFERENCES user(id),
        lat FLOAT,
        longitude FLOAT
    );

-- COMPANY: company_id,company_name,phone,email,country,website
    CREATE TABLE IF NOT EXISTS company (
        id VARCHAR(15) PRIMARY KEY,
        company_name VARCHAR(255),
        phone VARCHAR(15),
        email VARCHAR(100),
        country VARCHAR(100),
        website VARCHAR(255)
    );
```

```sql

-- PRODUCTS: id, product_name,price,colour,weight,warehouse_id
    CREATE TABLE IF NOT EXISTS products (
        id VARCHAR(15) PRIMARY KEY,
        product_name VARCHAR(255),
        price VARCHAR(1000),
        color VARCHAR(15),
        weight DECIMAL(10, 2),
        warehouse_id VARCHAR(15)
    );

    SHOW COLUMNS FROM products;
```

100% 1:30

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
product_name	varchar(255)	YES		NULL	
price	varchar(1000)	YES		NULL	
color	varchar(15)	YES		NULL	
weight	decimal(10,2)	YES		NULL	
warehouse_id	varchar(15)	YES		NULL	

```sql
-- CREDIT_CARD: id,user_id,iban,pan,pin,cvv,track1,track2,expiring_date
    CREATE TABLE IF NOT EXISTS credit_card (
        id VARCHAR(255) PRIMARY KEY NOT NULL,
        user_id CHAR(10),
        iban VARCHAR(50),
        pan VARCHAR(255),
        pin INT(10),
        cvv INT(10),
        track1 VARCHAR (255),
        track2 VARCHAR (255),
        expiring_date VARCHAR(255)
);
```

100% 34:80

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
iban	varchar(50)	YES		NULL	
pan	varchar(255)	YES		NULL	
pin	int(10)	YES		NULL	
cvv	int(10)	YES		NULL	
track1	varchar(255)	YES		NULL	
track2	varchar(255)	YES		NULL	

Data load:

➔ Primero intenté subir los archivos con el comando LOAD DATA LOCAL INFILE, siguiendo el tutorial para solventar el error --secure-file-priv; pero luego de varios intentos tanto en MySQL y iOS terminal; no fue posible.

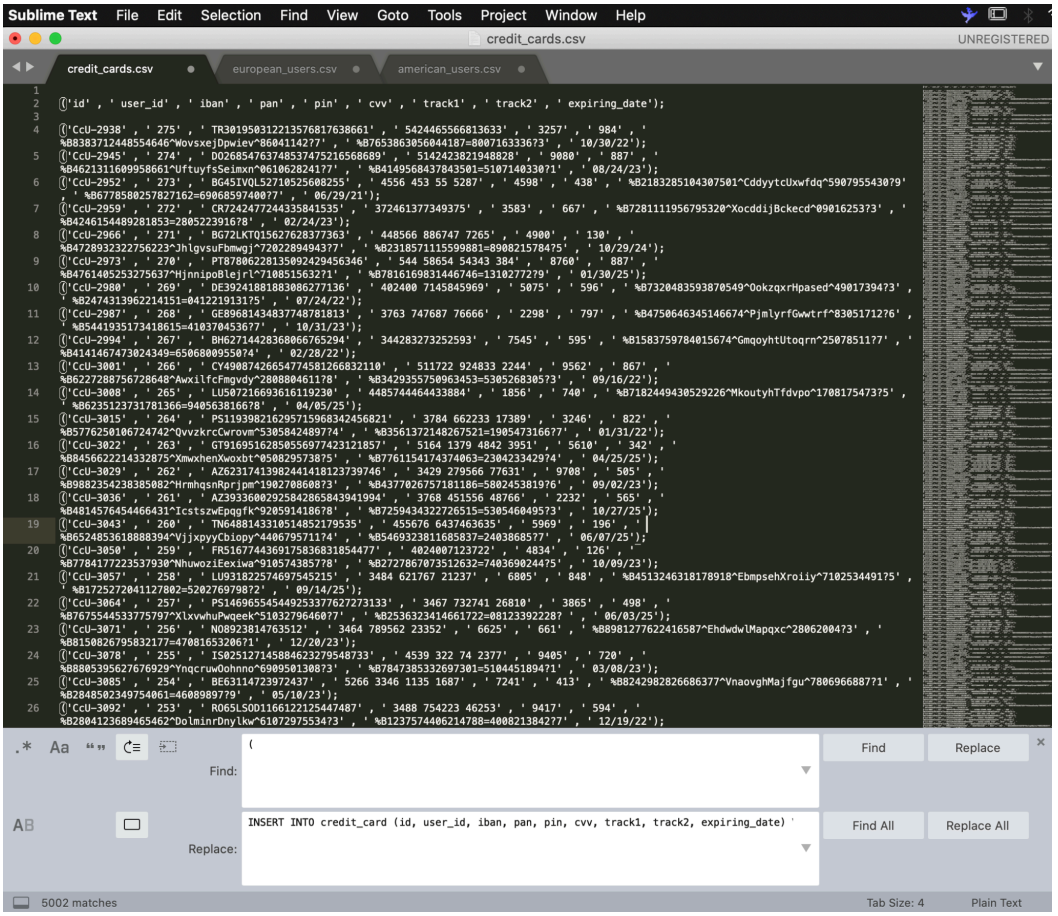
```
124    -- LOAD DATA
125
126 •      LOAD DATA LOCAL INFILE '/Users/minu/[PROJECT]/_ 2025/ 05_ Data Analysis/_ SPRINT 4/transactions.csv'
127        INTO TABLE transaction;
128        #Error Code: 1148. The used command is not allowed with this MySQL version
129
130 •      SHOW VARIABLES LIKE 'local_infile';
131        # output: local_infile, OFF
132
133 •      SET GLOBAL local_infile = 1;
134 •      SHOW VARIABLES LIKE 'local_infile';
135        #output: local_infile, ON
136
137 •      LOAD DATA LOCAL INFILE '~/dumps/transactions.csv'
138        INTO TABLE transaction;
139        #Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on access.
140
141 •      SHOW VARIABLES LIKE 'secure_file_priv';
142        #output: secure_file_priv, NULL
```

100% 1:122

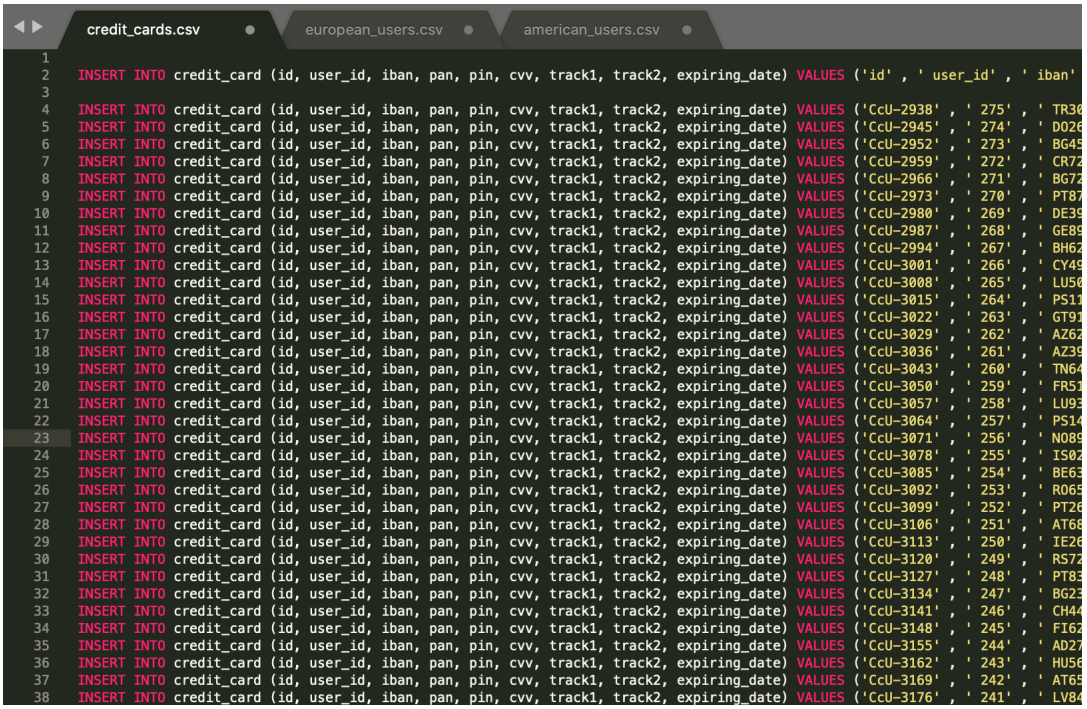
Result Grid Filter Rows: Search Export:

Variable_name	Value
secure_file_priv	NULL

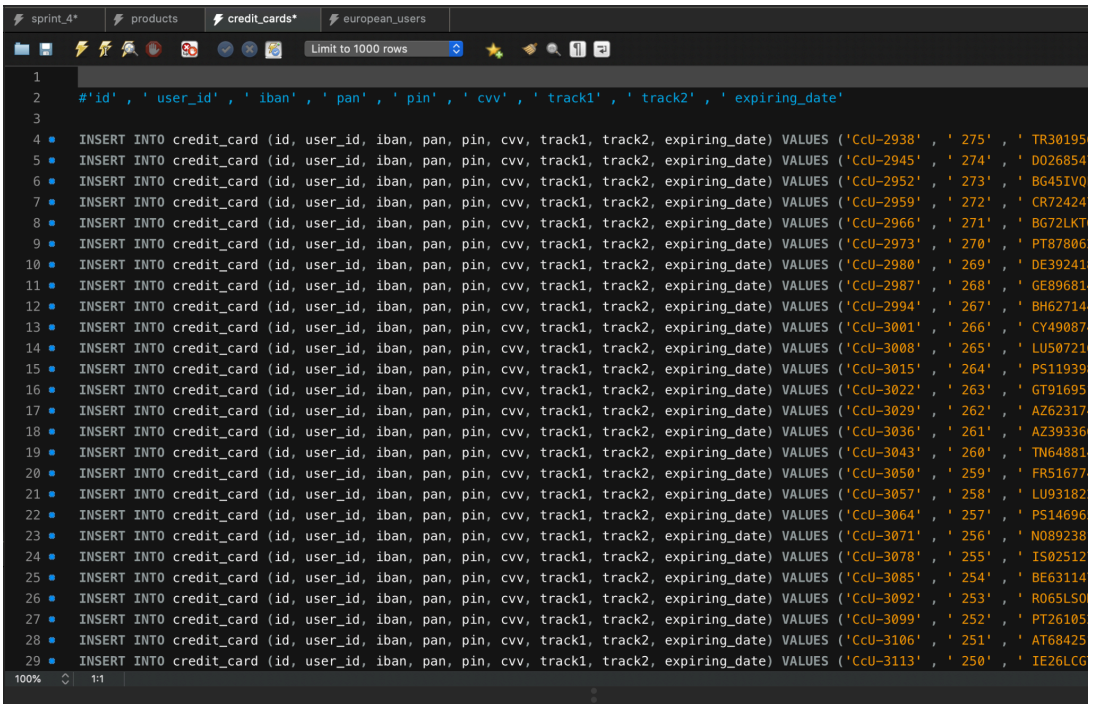
➔ Preparé los archivos manualmente. Realicé data cleaning de los archivos cvs. Edición de los datos “crudos” para que sean usables (edición del formato, por syntaxis sql).



➔ Agrego el comando INSERT y exporto el archivo formato sql.



➔ Abro los archivos sql en MySQL. Ejecutamos los INSERT INTO desde los archivos que hemos creado.



```
88
89 •      SELECT * FROM credit_card;
90
```

100% 32:83

Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

id	user_id	iban	pan	pin	cvv	track1	track2	expiring_date
CcS-4857	276	XX4857591835292505850771	2314242385113924	1819	467	%B2314242385113924^LWCBUDLWCBUD^22...	%B2314242385113924=2410101518363164?	09/27/25
CcS-4858	277	XX8581768137002436094025	6582720299715533	3964	817	%B6582720299715533^TIQMVITIQMVI^24040...	%B6582720299715533=2411011046546272?	12/28/28
CcS-4859	278	XX7826930491423553609370	8861684536289642	4983	277	%B8861684536289642^COFBGDCOFBGD^28...	%B8861684536289642=2502101761665371?	11/26/26
CcS-4860	279	XX5559590368835304645299	2481155515498459	6876	661	%B2481155515498459^TIUJTUTIUJTU^31040...	%B2481155515498459=2602101514414395?	07/27/27
CcS-4861	280	XX2035182877195191627307	1308930301149557	5710	398	%B1308930301149557^HPOBNZHPOBNZ^33...	%B1308930301149557=2805101751305028?	04/25/26
CcS-4862	281	XX4774721462463645409758	6715617009807829	4042	174	%B6715617009807829^LDMWTDLDMWTD^3...	%B6715617009807829=2210101702370428?	11/27/26
CcS-4863	282	XX1476829664245046207111	3140879819451394	5969	449	%B3140879819451394^OXXJODOXXJOD^23...	%B3140879819451394=3210101158648599?	12/27/29
CcS-4864	283	XX8380298893385731196159	5793672133649114	8481	139	%B5793672133649114^NHWBYRNHWBYR^3...	%B5793672133649114=2306101806367101?	02/28/26
CcS-4865	284	XX7085078596101025280599	5101552687251312	7847	903	%B5101552687251312^MJODHKMJODHK^33...	%B5101552687251312=3010101664862710?	11/25/28
CcS-4866	285	XX4792859188206596406839	8080768801072613	9271	961	%B8080768801072613^MEBGOZMEBGOZ^3...	%B8080768801072613=2810101715885695?	02/28/25
CcS-4867	286	XX6038298816319374853717	7761849537661098	4820	862	%B7761849537661098^LDRSOLLDRSOL^281...	%B7761849537661098=2206101710411371?	11/30/25
CcS-4868	287	XX3929101617300533068044	4160419663151801	6860	549	%B4160419663151801^KQUCOIKQUCOI^270...	%B4160419663151801=2508101278587927?	08/25/25
CcS-4869	288	XX1566712096111531886465	5892930716233310	4775	448	%B5892930716233310^DNSCAVDNSCAV^25...	%B5892930716233310=2403101253758128?	09/29/25
CcS-4870	289	XX6443663804167732133949	2227240879956178	5872	715	%B2227240879956178^NNZJTRNNZJTR^340...	%B2227240879956178=3411101430032222?	01/25/25
CcS-4871	290	XX3520209528376786066563	6944143455556838	7992	643	%B6944143455556838^FRFEEUFRFEEU^270...	%B6944143455556838=3212101363716238?	07/26/26
CcS-4872	291	XX4091621143155871546334	5035418997919459	3642	191	%B5035418997919459^ISXYXZISXYXZ^3308...	%B5035418997919459=2812101790314055?	12/29/26
CcS-4873	292	XX1362497434153372311591	3270329845604923	1038	177	%B3270329845604923^XQTOZDXQTOZD^28...	%B3270329845604923=3112101515908414?	04/25/26
CcS-4874	293	XX9111642734917968970741	5388631654216752	2445	846	%B5388631654216752^UTUPBJUTUPBJ^340	%B5388631654216752=3411101437145559?	09/30/27

credit_card 4

Action Output

	Time	Action	Response
4	10:21:49	SELECT * FROM credit_card LIMIT 0, 1000	1000 row(s) returned

➔ Comprobamos que los datos se hayan subido correctamente.
➔ Esto lo repetimos con las 4 tablas.

➔ Una vez subido los datos, empiezo a crear las PRIMARY y FOREIGN KEY. Me encuentro varios errores, que hay que verificar y modificar **data_types.**

```
29 •      ALTER TABLE transaction MODIFY COLUMN user_id VARCHAR(255);
```

130% 1:29

Action Output

	Time	Action	Response
10...	11:52:36	ALTER TABLE transaction MODIFY COLUMN user_id VARCHAR(255)	100000 row(s) affected

```
83
84        #Agrego la primary key nuevamente
85 •      ALTER TABLE products ADD PRIMARY KEY (id);
86
```

42:82

Action Output

	Time	Action
1076	12:08:34	ALTER TABLE products ADD

```
73 •      SHOW COLUMNS FROM products;
74 •      SELECT * FROM products;
```

1:72

Result Grid | Filter Rows: Search | Export:

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
product_name	varchar(255)	YES		NULL	
price	varchar(1000)	YES		NULL	
color	varchar(15)	YES		NULL	
weight	decimal(10,2)	YES		NULL	
warehouse_id	varchar(15)	YES		NULL	

➔ **Tabla user,** unificaré los datos de american_user y european_user; pero mantendré esa distinción agregando datos en una nueva columna “region” (creada al inicio). Primero subo los datos european y agrego el registro ‘eu’ para la columna region. Luego hago lo mismo con los datos de american_user (usa).

```
142
143     -- LOAD DATA archivos cvs editados y transformados en sql.
144
145         #USER
146         SELECT * FROM user;
147
148             #soluciones conflictos al subir datos european_users en user
149             SELECT * FROM user
150             WHERE id IS NULL;
151
152             DELETE FROM user
153             WHERE id IS NULL;
154
155             #agrego dato 'eu' en la columna region:
156             UPDATE user SET region='eu' WHERE region IS NULL;
157
158             #Load american_users y dato "usa" en columna region:
159             UPDATE user SET region='usa' WHERE region IS NULL;
160
```

100% 25:141

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

id	name	surname	phone	email	birth_date	country	city	postal_code	address	region
1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30 1984	United States	San Jose	95101	P.O. Box 773	usa
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca	Sep 15 1989	United States	San Jose	95101	Ap #644-8492 Sagittis St.	usa
1000	Amkjrv	Qbulrxbp	+48-258-9936	amkjrv.qbulrxbp@example.com	May 17 1970	Germany	Stuttgart	70173	215 Qbulrxbp St	eu
1001	Nfvrlb	Oydaiwbg	+94-121-2522	nfvrlb.oydaiwbg@example.com	Mar 4 1994	Germany	Cologne	50667	121 Oydaiwbg St	eu
1002	Ijbfmd	Jbddzhvp	+70-120-3668	ijbfmd.jbddzhvp@example.com	Sep 27 2001	Germany	Munich	80331	412 Jbddzhvp St	eu
1003	Uyciig	Sfbdymzj	+58-123-6968	uyciig.sfbdymzj@example.com	Jan 20 1981	Germany	Stuttgart	70173	735 Sfbdymzj St	eu

user 10

Action Output

	Time	Action	Response
11	16:52:05	SELECT * FROM user LIMIT 0, 1000	1000 row(s) returned

➔ **Tabla transacción.** Al crear las **FK (**transaction.product_id con product.id) el resultado **da error**.

```
214
215     #FK transaction + products:
216         ALTER TABLE transaction MODIFY COLUMN product_id VARCHAR(255);
217         ALTER TABLE transaction ADD FOREIGN KEY (product_id) REFERENCES products(id);
218         #Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`sprint_4`.`#sql-a3_d`, CONSTRAINT
```

100% 1:213

Action Output

	Time	Action	Response
3	15:49:16	ALTER TABLE transaction ADD FOREIGN KEY (product_id) REFERENCES products(id)	Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`spr...

Hago el comando WHERE .. NOT IN, pensando que tal vez es un registro huérfano. Pero al ver los **múltiples valores**, pues es esto lo que hay que solucionar.

```
146
147 •       SELECT product_id
148         FROM transaction
149         WHERE product_id NOT IN (SELECT id FROM products)
150
```

2:141 | 1 error found

Result Grid | Filter Rows: Search | Export: | Fetch rows:

product_id
30, 11, 16, 81
35, 33, 19
93, 55, 28, 91
55, 8, 72
46, 56, 73
6, 89, 19
58, 61, 55
8, 91
76, 19, 89
54, 17
88, 4
52, 3, 91, 75

transaction 40

Action Output

	Time	Action	Response
10...	13:16:15	SELECT product_id FROM transaction WHERE product_id NOT IN (SELECT...	1000 row(s) returned

Esto significa que tenemos que ***normalizar la tabla***: solucionar el conflicto de múltiples valores en una celda (primera norma formal: valores atómicos) y así resolver la imposibilidad de usar FOREIGN KEY. Para eso hay que crear una **tabla de relación/unión N:M**.

➔ Creación de tabla de relación ***transaction_products***

+

➔ Extracción de los datos múltiples ('1, 12, 86' en t.product_id): convertirlos en filas individuales.

```
199     -- crear nueva tabla separando los productos en cada celda. #JSON
200 •       CREATE TABLE transaction_products AS
201         SELECT
202             transaction.id AS transaction_id,
203             jt.product AS product_id
204         FROM transaction
205             JOIN JSON_TABLE(
206             CONCAT('[', transaction.product_id, ']'),
207             '$[*]' COLUMNS (product VARCHAR(10) PATH '$')
208         ) AS jt;
209
210 •       SELECT * FROM transaction_products;
```

1:198

Result Grid | Filter Rows: Search | Export: | Fetch rows:

transaction_id	product_id
00043A49-2949-494B-A5DD-A5BAE3BB19DD	97
00043A49-2949-494B-A5DD-A5BAE3BB19DD	87
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	66
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	69
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	87
00045D6B-ED2E-4F2F-8186-CEE074D875D0	30
00045D6B-ED2E-4F2F-8186-CEE074D875D0	11
00045D6B-ED2E-4F2F-8186-CEE074D875D0	16
00045D6B-ED2E-4F2F-8186-CEE074D875D0	81
000481C3-1C26-4FEF-83A0-4CD0EB004BBD	72
00051AA4-9CBE-4268-B070-C38062A1B3E2	18
0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	35
0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	33
0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	19

transaction_products 61

Action Output

	Time	Action	Response
111	13:40:35	SELECT * FROM transaction_products LIMIT 0, 1000	1000 row(s) returned

➔ Añadir FK (las configuramos luego de tener toda la data en la nueva tabla)
transaction_id → transaction(id)
product_id → products(id).

```
242
243             #FK transaction + transaction_products:
244 •               ALTER TABLE transaction_products ADD FOREIGN KEY(transaction_id) REFERENCES transaction(id);
245 •               ALTER TABLE transaction_products ADD FOREIGN KEY (product_id) REFERENCES products(id);
246
247 •               SHOW COLUMNS FROM transaction_products;
```

5:229

Result Grid Filter Rows: Search Export:

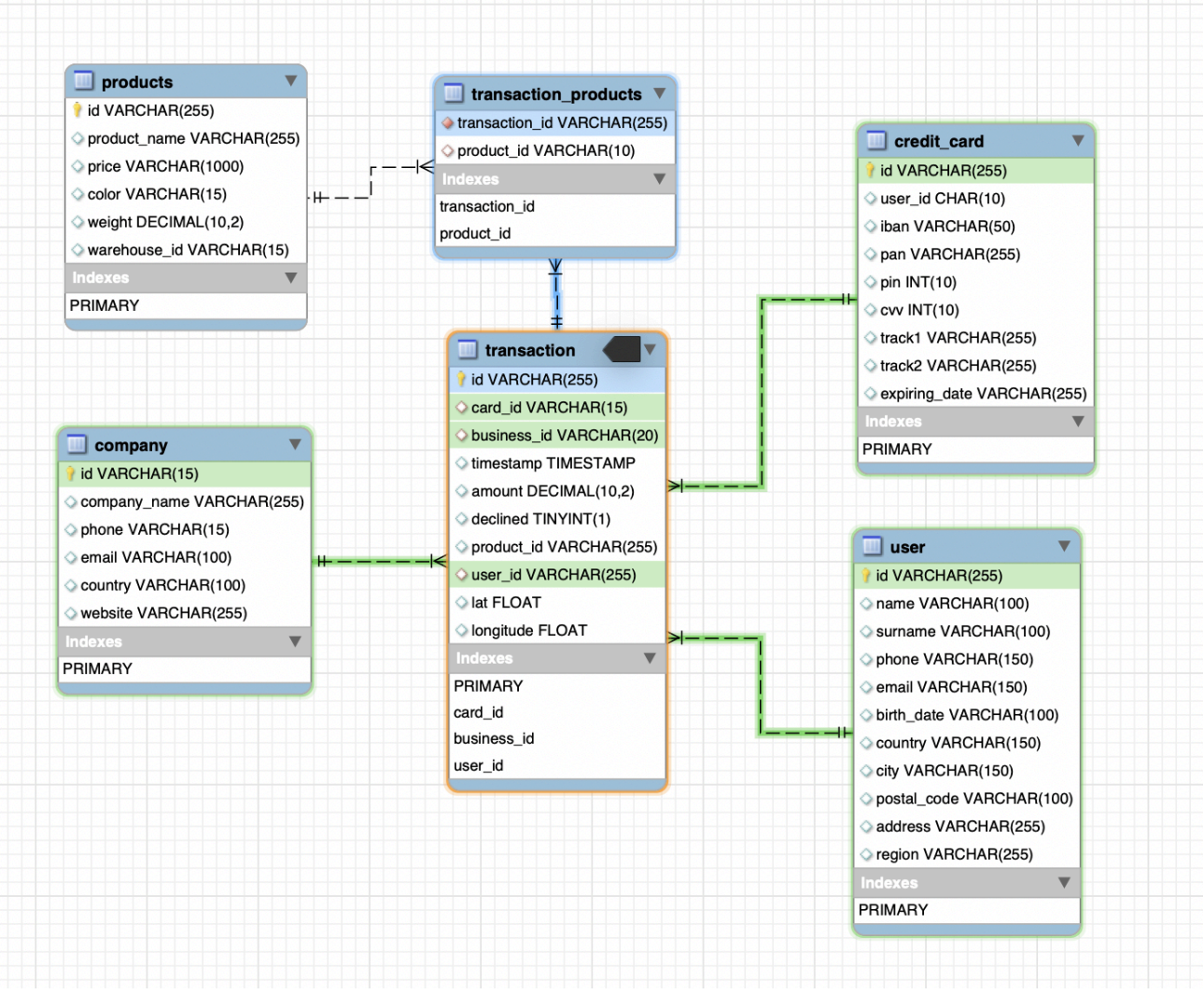
Field	Type	Null	Key	Default	Extra
transaction_id	varchar(255)	NO	MUL	NULL	
product_id	varchar(10)	YES	MUL	NULL	

Result 4

Action Output

	Time	Action	Response
5	16:03:26	SHOW COLUMNS FROM transaction_products	2 row(s) returned

Finalmente con todas las tablas creadas, con los datos y las FK creadas, el **diagrama** ha quedado así.
Podríamos decir que por la relación en que han quedado las tablas:
transaction -> transaction_products <- products, ha quedado un diagrama de ***copo de nieve***.



NIVEL 1 |

Ejercicio 1

*Realiza una subconsulta que muestre a todos los **usuarios** con **más de 80 transacciones** utilizando al menos 2 tablas.*
#Lógica: contar total transactions, agrupar por user_id.

```
252    SELECT t.id, t.total_transactions
253    FROM (
254        SELECT user.id,
255        COUNT(transaction.id) AS total_transactions
256        FROM user
257        JOIN transaction
258            ON transaction.user_id=user.id
259        GROUP BY user.id
260    ) AS t
261    WHERE t.total_transactions > 80
262    ORDER BY total_transactions;
```

100% 1:248

Result Grid Filter Rows: Search Export:

id	total_transactio...
454	81
318	91

NIVEL 1 |

Ejercicio 2

*Muestra la **media de amount por IBAN** de las tarjetas de crédito en la compañía **Donec Ltd.**, utiliza por lo menos 2 tablas.*

```
319    SELECT
320        cc.iban,
321        AVG(t.amount) AS promedio
322    FROM credit_card cc
323    JOIN transaction t
324        ON t.card_id=cc.id
325    WHERE t.business_id = (
326        SELECT id
327        FROM company
328        WHERE company_name = 'Donec Ltd'
329        )
330    GROUP BY cc.iban
331    ORDER BY promedio DESC;
```

1:318

Result Grid Filter Rows: Search Export:

iban	promedio
XX3830178139196201993663 52	680.690000
XX6377063573975703949739 13	680.010000
XX9713939714652922023122 59	645.460000
XX1718471169288923759693 07	628.890000
XX2254246388185424062235 75	608.680000
XX7488907290571957117660 71	607.290000
TN9614563570667381893122	605.410000
XX4819080340373642425911 85	605.360000
XX1946755197392563357535 08	597.190000
XX2159627660619671954934 37	594.260000
XX4493223208268907210014 43	591.610000
XX5351854927357042294742 37	570.090000
CH9552373968796160224	566.380000
XX3476053771256378803031 31	561.800000

Result 52

Action Output

	Time	Action	Response
89	13:00:41	SELECT cc.iban, AVG(t.amount) AS promedio FR...	371 row(s) returned

NIVEL 2 |

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si ***las tres últimas transacciones han sido declinadas entonces es inactivo****, si al menos* ***una no es rechazada entonces es activo*** .

➔ Armo la consulta

```
WITH last_3 AS (
        SELECT
            card_id, declined,
            ROW_NUMBER () OVER (
                PARTITION BY card_id
                ORDER BY timestamp DESC
                ) AS rn
        FROM transaction
        )
    SELECT
        card_id,
        CASE
            WHEN SUM(declined) = 3 THEN 'inactive'
            ELSE 'active'
        END AS status
    FROM last_3
    WHERE rn <= 3
    GROUP BY card_id;
```

100% 4:326

Result Grid | Filter Rows: Search | Export: | Fetch rows:

card_id	status
CcS-4857	active
CcS-4858	active
CcS-4859	active
CcS-4860	active
CcS-4861	active
CcS-4862	active
CcS-4863	active

Result 1

Action Output

	Time	Action	Response
1	11:01:29	WITH last_3 AS (SELECT card_id, declined, ROW_NUMBER () OVER (PARTITION BY card_id O...	5000 row(s) returned

➔ Creo la tabla

```
CREATE TABLE card_status AS
  WITH last_3 AS (
        SELECT
            card_id, declined,
            ROW_NUMBER () OVER (
                PARTITION BY card_id
                ORDER BY timestamp DESC
                ) AS rn
        FROM transaction
        )
    SELECT
        card_id,
        CASE
            WHEN SUM(declined) = 3 THEN 'inactive'
            ELSE 'active'
        END AS status
    FROM last_3
    WHERE rn <= 3
    GROUP BY card_id;
```

100% 2:327

Action Output

	Time	Action	Response
2	11:05:06	CREATE TABLE card_status AS WITH last_3 AS (SELECT card_id, declined, ROW_NUMBER () O...	5000 row(s)

NIVEL 2 |

Ejercicio 1

¿Cuántas tarjetas están activas?

➔ Respuesta: **4995** tarjetas activas

```
354     -- NIVEL 2: Ejercicio 1
355     SELECT COUNT(status)
356     FROM card_status
357     WHERE status = 'active';
358
```

100% 17:348

Result Grid | Filter Rows: Search

COUNT(status)
4995

NIVEL 3

*Crea una tabla con la que podamos unir los datos del nuevo archivo **products.csv** con la base de datos creada, teniendo en cuenta que desde **transaction** tienes **product_ids**. Genera la siguiente consulta:*

➔ Hemos creado la tabla nueva (transaction_products) al inicio al crear la base de datos, subir todos los datos y la creación del diagrama.

NIVEL 3 |

Ejercicio 1

Necesitamos conocer el número de veces que se ha vendido cada producto.

```
296
297     SELECT
298         p.id AS product_id,
299         p.product_name,
300         COUNT(t.product_id) AS total_sold
301     FROM products p
302     JOIN transaction_products t ON p.id=t.product_id
303     GROUP BY
304         p.id,
305         p.product_name
306     ORDER BY product_id;
307
```

100% 17:294

Result Grid | Filter Rows: Search | Export:

product_id	product_name	total_sold
1	Direwolf Stannis	2468
10	Karstark Dorne	2571
100	south duel	2517
11	Karstark Dorne	2573
12	duel Direwolf	2558
13	palpatine chewbacca	2560
14	Direwolf	2496
15	Stannis warden	2535
16	the duel warden	2608

Result 18

Action Output

	Time	Action	Response
22	17:31:09	SELECT p.id AS product_id, p.product_name, COUNT(t.product_id) AS to...	100 row(s) returned