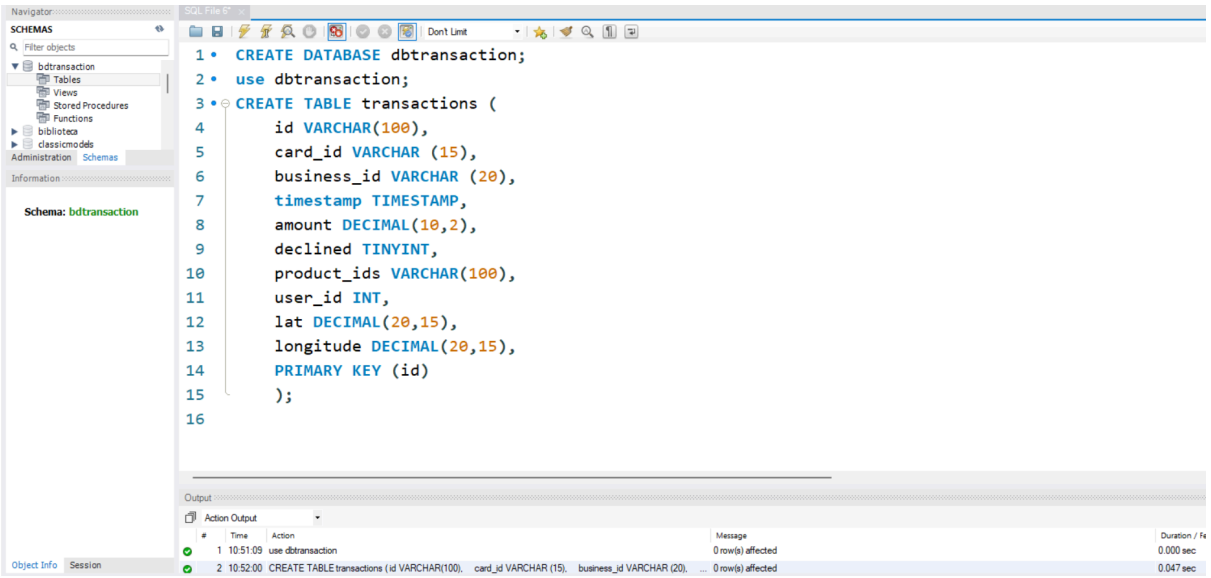
SPRINT 4

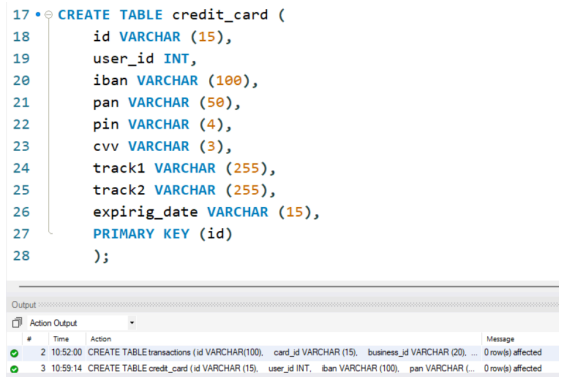
Nivel 1

Descarga los archivos CSV, estudiales 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:

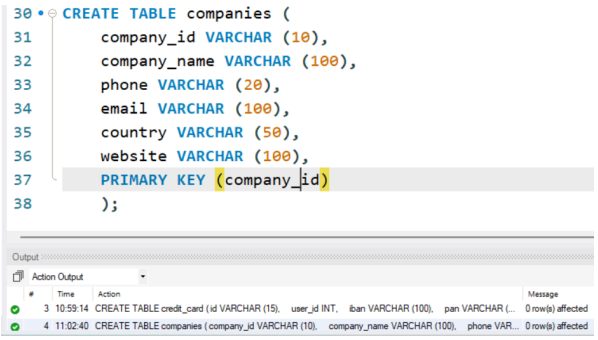
En primer lugar se crea la base de datos (dbtransaction), y dentro de ella las diferentes tablas: **transactions, credit_card, companies** y **users,** definiendo los tipos de variables y sus claves primarias**:**



```sql
CREATE DATABASE dbtransaction;
use dbtransaction;
CREATE TABLE transactions (
    id VARCHAR(100),
    card_id VARCHAR (15),
    business_id VARCHAR (20),
    timestamp TIMESTAMP,
    amount DECIMAL(10,2),
    declined TINYINT,
    product_ids VARCHAR(100),
    user_id INT,
    lat DECIMAL(20,15),
    longitude DECIMAL(20,15),
    PRIMARY KEY (id)
    );
```



```sql
CREATE TABLE credit_card (
    id VARCHAR (15),
    user_id INT,
    iban VARCHAR (100),
    pan VARCHAR (50),
    pin VARCHAR (4),
    cvv VARCHAR (3),
    track1 VARCHAR (255),
    track2 VARCHAR (255),
    expirig_date VARCHAR (15),
    PRIMARY KEY (id)
    );
```



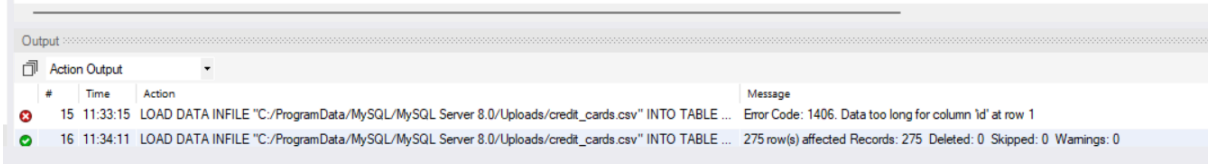
```sql
CREATE TABLE companies (
    company_id VARCHAR (10),
    company_name VARCHAR (100),
    phone VARCHAR (20),
    email VARCHAR (100),
    country VARCHAR (50),
    website VARCHAR (100),
    PRIMARY KEY (company_id)
    );
```

```sql
CREATE TABLE users (
    id INT,
    name VARCHAR (50),
    surname VARCHAR (50),
    phone VARCHAR (20),
    email VARCHAR (100),
    birth_date VARCHAR (15),
    country VARCHAR (50),
    city VARCHAR (50),
    postal_code VARCHAR (10),
    address VARCHAR (100),
    PRIMARY KEY (id)
    );
```

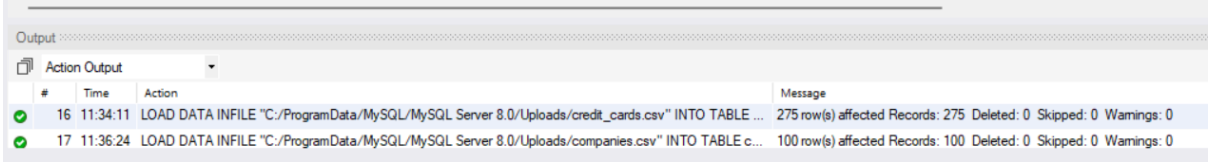
Continuamos con el volcado de datos en cada tabla mediante los comandos:

```sql
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```


```sql
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv"
INTO TABLE credit_card
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

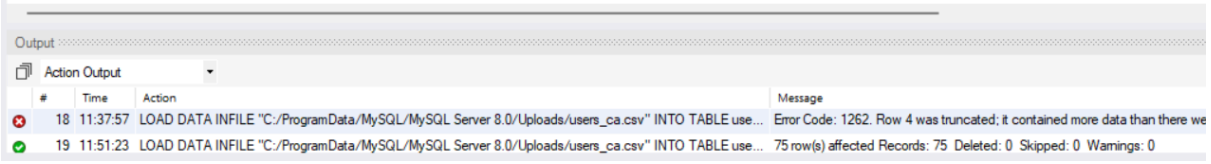



```sql
LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv"
INTO TABLE companies
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
ESCAPED BY '\\'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```

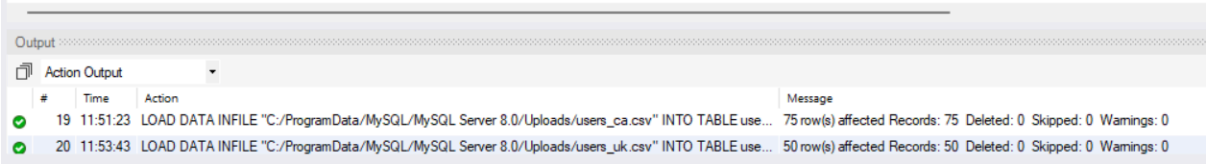


En el caso de la tabla **users** tenemos 3 archivos que volcar. Además surge la dificultad de que en el campo address, en algunos casos, el registro aparece entre comillas (“”), por lo que se ha de indicar, además de OPTIONALLY ENCLOSED BY ‘ “ ‘, la opción LINES TERMINATED BY ‘\r\n’.

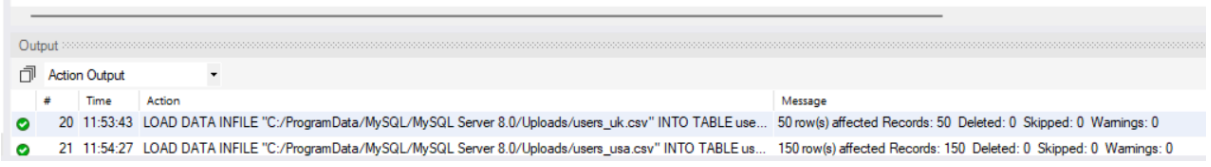
```
81 •  LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv"
82    INTO TABLE users
83    FIELDS TERMINATED BY ','
84    OPTIONALLY ENCLOSED BY '"'
85    ESCAPED BY '\\'
86    LINES TERMINATED BY '\r\n'
87    IGNORE 1 LINES;
```




```
89 •  LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv"
90    INTO TABLE users
91    FIELDS TERMINATED BY ','
92    OPTIONALLY ENCLOSED BY '"'
93    ESCAPED BY '\\'
94    LINES TERMINATED BY '\r\n'
95    IGNORE 1 LINES;
```



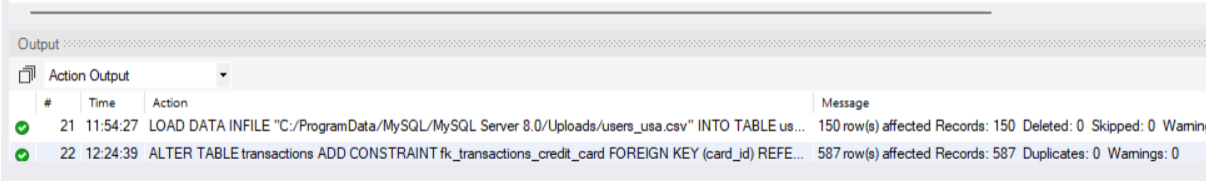

```
 98 •  LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv"
 99    INTO TABLE users
100    FIELDS TERMINATED BY ','
101    OPTIONALLY ENCLOSED BY '"'
102    ESCAPED BY '\\'
103    LINES TERMINATED BY '\r\n'
104    IGNORE 1 LINES;
```



Y seguidamente se crean las relaciones:

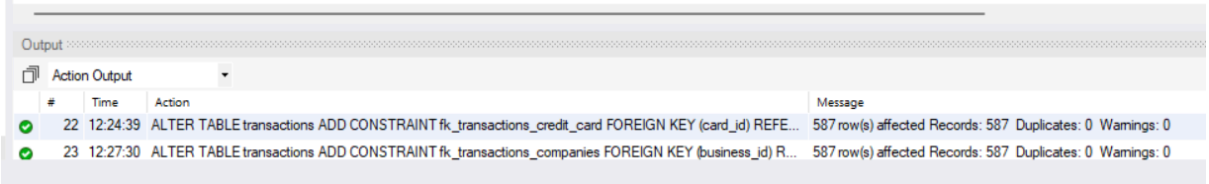
- Entre **transactions** y **credit_card:**

```
107 • ALTER TABLE transactions
108   ADD CONSTRAINT fk_transactions_credit_card
109   FOREIGN KEY (card_id) REFERENCES credit_card(id);
```



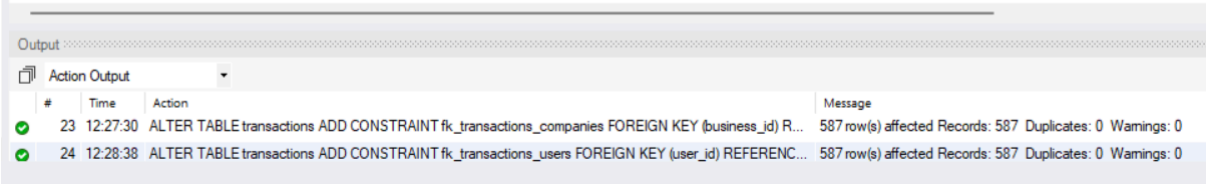
- Entre **transactions** y **companies:**

```
111 • ALTER TABLE transactions
112   ADD CONSTRAINT fk_transactions_companies
113   FOREIGN KEY (business_id) REFERENCES companies(company_id);
```

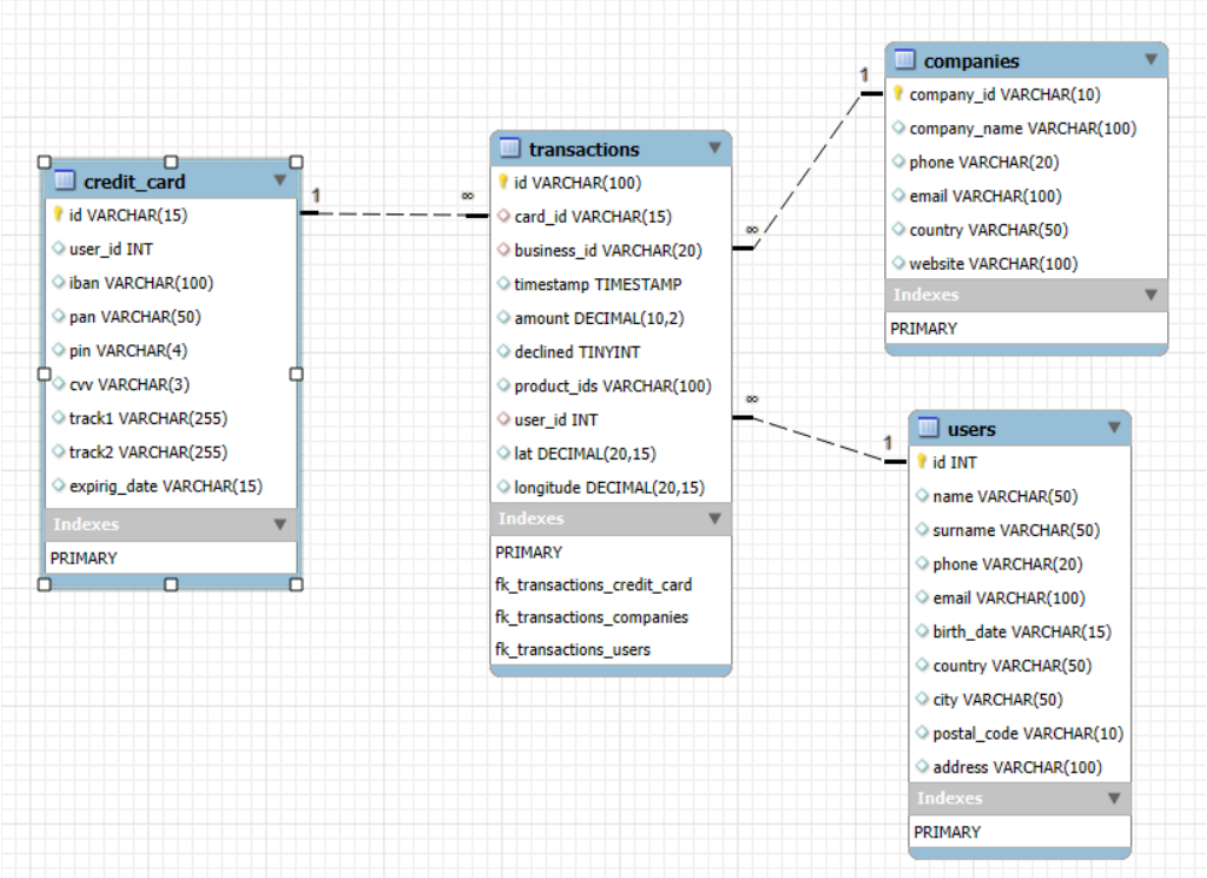


- Entre **transactions** y **users:**

```
115 • ALTER TABLE transactions
116   ADD CONSTRAINT fk_transactions_users
117   FOREIGN KEY (user_id) REFERENCES users(id);
```



Finalmente obtengo el diagrama de mi base de datos:



- Ejercicio 1

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

```sql
# Ejercicio 1.1:
SELECT u.id, u.name , u.surname, count(t.id) AS num_transacciones
FROM users u
JOIN transactions t
ON u.id = t.user_id
GROUP BY u.id, u.name , u.surname
HAVING num_transacciones > 30;
```

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

id	name	surname	num_transacciones
92	Lynn	Riddle	39
267	Ocean	Nelson	52
272	Hedwig	Gilbert	76
275	Kenyon	Hartman	48

Result 1

Output

Action Output

#	Time	Action	Message
27	12:22:01	use dbtransaction	0 row(s) affected
28	12:23:30	SELECT u.id, u.name , u.surname, count(t.id) AS num_transacciones FROM users u JOIN transactions t ON ...	4 row(s) returned

- 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.

```sql
# Ejercicio 1.2:
SELECT c.company_id, c.company_name, cc.iban, ROUND(AVG(t.amount), 3) AS gasto_medio
FROM companies c
JOIN transactions t
ON c.company_id = t.business_id
JOIN credit_card cc
ON t.card_id = cc.id
GROUP BY c.company_id, c.company_name, cc.iban
HAVING company_name = 'Donec Ltd';
```

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

company_id	company_name	iban	gasto_medio
b-2242	Donec Ltd	PT87806228135092429456346	203.715

Result 3

Output

Action Output

#	Time	Action	Message
29	12:24:39	SELECT c.company_id, c.company_name, cc.iban, AVG(t.amount) AS gasto_medio FROM companies c JOI...	1 row(s) returned
30	12:25:25	SELECT c.company_id, c.company_name, cc.iban, ROUND(AVG(t.amount), 3) AS gasto_medio FROM com...	1 row(s) returned

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:

```sql
CREATE TABLE card_status (
    card_id VARCHAR(15) PRIMARY KEY,
    estado VARCHAR(20)
);
```

Output

Action Output

#	Time	Action	Message
47	13:19:40	drop table card_status	0 row(s) affected
48	13:19:46	CREATE TABLE card_status (card_id VARCHAR(15) PRIMARY KEY, estado VARCHAR(20))	0 row(s) affected

```sql
INSERT INTO card_status (card_id, estado)
WITH ultimas_tres AS (
    SELECT
        card_id,
        declined,
        ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS rn
    FROM transactions),
conteo_declinadas AS (
    SELECT card_id, SUM(declined) AS declinadas
    FROM ultimas_tres
    WHERE rn <= 3
    GROUP BY card_id)
SELECT card_id,
    CASE
    WHEN declinadas = 3 THEN 'rechazada'
    ELSE 'activa'
    END AS estado
FROM conteo_declinadas;
```

Output

Action Output

#	Time	Action	Message
22	10:45:20	CREATE TABLE card_status (card_id VARCHAR(15), estado VARCHAR(20))	0 row(s) affected
23	10:46:23	INSERT INTO card_status (card_id, estado) WITH ultimas_tres AS (SELECT card_id, declined, ROW_...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

Compruebo la nueva tabla **card_status:**

```sql
SELECT * FROM card_status;
```

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

card_id	estado
CcU-2938	activa
CcU-2945	activa
CcU-2952	activa
CcU-2959	activa
CcU-2966	activa
CcU-2973	activa
CcU-2980	activa
CcU-2987	activa

card_status 4

Output

Action Output

#	Time	Action	Message
31	12:27:38	SELECT * FROM dbtransaction.card_status	275 row(s) returned
32	12:28:11	SELECT * FROM card_status	275 row(s) returned

Y añado la relación de esta tabla a la tabla **credit_card**:

```sql
ALTER TABLE card_status
ADD CONSTRAINT fk_credit_card_card_status
FOREIGN KEY (card_id) REFERENCES credit_card(id);
```

Output

Action Output

#	Time	Action	Message
45	13:02:35	SELECT p.id, p.product_name, COUNT(transaction_id) AS num_ventas FROM products p JOIN products_tr...	26 row(s) returned
46	13:08:59	ALTER TABLE card_status ADD CONSTRAINT fk_credit_card_card_status FOREIGN KEY (card_id) REFE...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

Ejercicio 1

¿Cuántas tarjetas están activas?

```
166 •  SELECT COUNT(card_id)
167    FROM card_status
168    WHERE estado = 'activa';
169
```

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

COUNT(card_id)
275

Result 2

Output

Action Output

#	Time	Action	Message
23	10:46:23	INSERT INTO card_status (card_id, estado) WITH ultimas_tres AS (SELECT card_id, declined, ROW_...	275 row(s) affected F
24	10:48:35	SELECT * FROM card_status WHERE estado = 'activa'	275 row(s) returned
25	10:49:25	SELECT COUNT(card_id) FROM card_status WHERE estado = 'activa'	1 row(s) returned

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:

En primer lugar creo la tabla **products:**

```
174 •  CREATE TABLE products (
175        id INT PRIMARY KEY,
176        product_name VARCHAR (150),
177        price VARCHAR (50),
178        colour VARCHAR (50),
179        weight DECIMAL (5,1),
180        warehouse_id VARCHAR (20)
181        );
182
```

Output

Action Output

#	Time	Action	Message
24	10:48:35	SELECT * FROM card_status WHERE estado = 'activa'	275 row(s) returned
25	10:49:25	SELECT COUNT(card_id) FROM card_status WHERE estado = 'activa'	1 row(s) returned
26	11:41:43	CREATE TABLE products (id INT PRIMARY KEY, product_name VARCHAR (150), price VARCHAR (5...	0 row(s) affected

Y vuelco los datos del archivo .csv en ella:

```
183 •  LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
184    INTO TABLE products
185    FIELDS TERMINATED BY ','
186    OPTIONALLY ENCLOSED BY '"'
187    ESCAPED BY '\\'
188    LINES TERMINATED BY '\n'
189    IGNORE 1 LINES;
```

Output

Action Output

#	Time	Action	Message
26	11:41:43	CREATE TABLE products (id INT PRIMARY KEY, product_name VARCHAR (150), price VARCHAR (5...	0 row(s) affected
27	11:48:28	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' INTO TABLE trans...	Error Code: 1406. Data too long for column 'card_id' at row 1
28	11:48:49	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' INTO TABLE prod...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Compruebo que la carga se ha realizado correctamente:

```
191 •  SELECT * FROM products
```

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

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	$161.11	#7c7c7c	1.0	WH-4
2	Tarly Stark	$9.24	#919191	2.0	WH-3
3	duel tourney Lannister	$171.13	#d8d8d8	1.5	WH-2
4	warden south duel	$71.89	#111111	3.0	WH-1
5	skywalker ewok	$171.22	#dbdbdb	3.2	WH-0

products 3

Output

Action Output

#	Time	Action	Message
28	11:48:49	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' INTO TABLE prod...	100 row(s) affected Reco
29	11:49:51	SELECT * FROM products	100 row(s) returned

Ahora procedo a cambiar el tipo de dato del campo *precio*, para que sea un decimal, ya que el formato que me viene dado incluye el símbolo $, lo cual me impide tratarlo como un numérico y me ha obligado a darle el tipo varchar a esa columna. Para ello, en primer lugar creo una nueva columna con el precio numérico de tipo decimal: *precio_num.*

```
193 •  ALTER TABLE products ADD COLUMN precio_num DECIMAL(10,2);
194
```

Output

Action Output

#	Time	Action	Message
29	11:49:51	SELECT * FROM products	100 row(s) returned
30	11:53:37	ALTER TABLE products ADD COLUMN precio_num DECIMAL(10,2)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Después la relleno con los datos de la columna *precio* pero eliminando el símbolo $:

```
195 •  UPDATE products
196    SET precio_num = CAST(REPLACE(price, '$', '') AS DECIMAL(10,2));
```

Output

Action Output

#	Time	Action	Message
1	12:01:46	use dbtransaction	0 row(s) affected
2	12:03:38	UPDATE products SET precio_num = CAST(REPLACE(price, '$', '') AS DECIMAL(10,2))	100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0

Compruebo los cambios.

```
191 •  SELECT * FROM products;
192
```

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

id	product_name	price	colour	weight	warehouse_id	precio_num
1	Direwolf Stannis	$161.11	#7c7c7c	1.0	WH-4	
2	Tarly Stark	$9.24	#919191	2.0	WH-3	9.24
3	duel tourney Lannister	$171.13	#d8d8d8	1.5	WH-2	171.13
4	warden south duel	$71.89	#111111	3.0	WH-1	71.89
5	skywalker ewok	$171.22	#dbdbdb	3.2	WH-0	171.22
6	dooku solo	$136.60	#c4c4c4	0.8	WH--1	136.60
7	north of Casterly	$63.33	#b7b7b7	0.6	WH--2	63.33
8	Winterfell	$32.37	#383838	1.4	WH--3	32.37
9	Winterfell	$76.40	#b5b5b5	1.2	WH--4	76.40
10	Karstark Dorne	$119.52	#f4f4f4	2.4	WH--5	119.52

products 1 | Apply

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	12:03:38	UPDATE products SET precio_num = CAST(RE...	100 row(s) affected Rows matched: 100 Change...	0.000 sec
3	12:06:39	SELECT * FROM products	100 row(s) returned	0.016 sec / 0.0

Y finalmente elimino la columna original *price* para no tener el precio duplicado. Y cambio el nombre de la columna *precio_num* a simplemente *price*.

```
198 •  ALTER TABLE products DROP COLUMN price;
199
200 •  ALTER TABLE products RENAME COLUMN precio_num TO price;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetc
4	12:10:06	ALTER TABLE products DROP COLUMN price	0 row(s) affected Records: 0 Duplicates: 0 War...	0.015 sec
5	12:11:11	ALTER TABLE products RENAME COLUMN pre...	0 row(s) affected Records: 0 Duplicates: 0 War...	0.016 sec

Compruebo los cambios:

```
191 •  SELECT * FROM products;
```

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

id	product_name	colour	weight	warehouse_id	price
1	Direwolf Stannis	#7c7c7c	1.0	WH-4	161.11
2	Tarly Stark	#919191	2.0	WH-3	9.24
3	duel tourney Lannister	#d8d8d8	1.5	WH-2	171.13
4	warden south duel	#111111	3.0	WH-1	71.89
5	skywalker ewok	#dbdbdb	3.2	WH-0	171.22
6	dooku solo	#c4c4c4	0.8	WH--1	136.60
7	north of Casterly	#b7b7b7	0.6	WH--2	63.33
8	Winterfell	#383838	1.4	WH--3	32.37
9	Winterfell	#b5b5b5	1.2	WH--4	76.40
10	Karstark Dorne	#f4f4f4	2.4	WH--5	119.52

products 2 | Apply

Output

Action Output

#	Time	Action	Message	Duration / Fetch
5	12:11:11	ALTER TABLE products RENAME COLUMN pre...	0 row(s) affected Records: 0 Duplicates: 0 War...	0.016 sec
6	12:12:13	SELECT * FROM products	100 row(s) returned	0.000 sec / 0.000

Ahora que tengo la tabla **products** creada correctamente, procedo a relacionarla con **transactions.** Para ello crearé una tabla intermedia **products_transactions**, ya que la relación entre ambas es de muchos a muchos, pues cada transacción puede incluir muchos productos, y cada producto puede estar incluido en múltiples transacciones.

```
205 • CREATE TABLE products_transactions (
206     transaction_id VARCHAR(100),
207     product_id INT,
208     PRIMARY KEY (transaction_id, product_id),
209     FOREIGN KEY (transaction_id) REFERENCES transactions(id),
210     FOREIGN KEY (product_id) REFERENCES products(id)
211 );
212
```

Output

Action Output

#	Time	Action	Message
39	12:56:06	drop table products_transactions	0 row(s) affected
40	12:56:50	CREATE TABLE products_transactions (transaction_id VARCHAR(100), product_id INT, PRIMARY KE...	0 row(s) affected

El siguiente paso es rellenar esta nueva tabla con los registros de **transaccions** y **products.** Para ello se ha de utilizar la función FIND_IN_SET:

```
213 • INSERT INTO products_transactions (transaction_id, product_id)
214   SELECT
215     t.id AS transaction_id,
216     p.id AS product_id
217   FROM transactions t
218   JOIN products p
219     ON FIND_IN_SET(p.id, REPLACE(t.product_ids, ' ', '')) > 0;
220
```

Output

Action Output

#	Time	Action	Message
40	12:56:50	CREATE TABLE products_transactions (transaction_id VARCHAR(100), product_id INT, PRIMARY KE...	0 row(s) affected
41	12:57:55	INSERT INTO products_transactions (transaction_id, product_id) SELECT t.id AS transaction_id, p.id AS ...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Ahora podemos comprobar el contenido de la tabla **products_transactions**:

```
222 • SELECT * FROM products_transactions;
223
```

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

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
122DC333-E19F-D629-DCD8-9C54CF1EBB9A	1
1753A288-9FC1-52E6-5C39-A1FFB97B0D3A	1
1A6CECFB-2E3A-65A3-72D9-2FDB58A1E4BA	1
1EA2B262-D507-AD14-4374-4D532967113F	1
23CF8ED3-402C-7C54-59CD-DB505C5CCCCE	1
2A5A3001-104F-1D1F-7852-5BA801869B6F	1
2F3B6AB6-147D-EB0B-FE8D-9A4E2EA9DBD5	1

products_transactions 6

Output

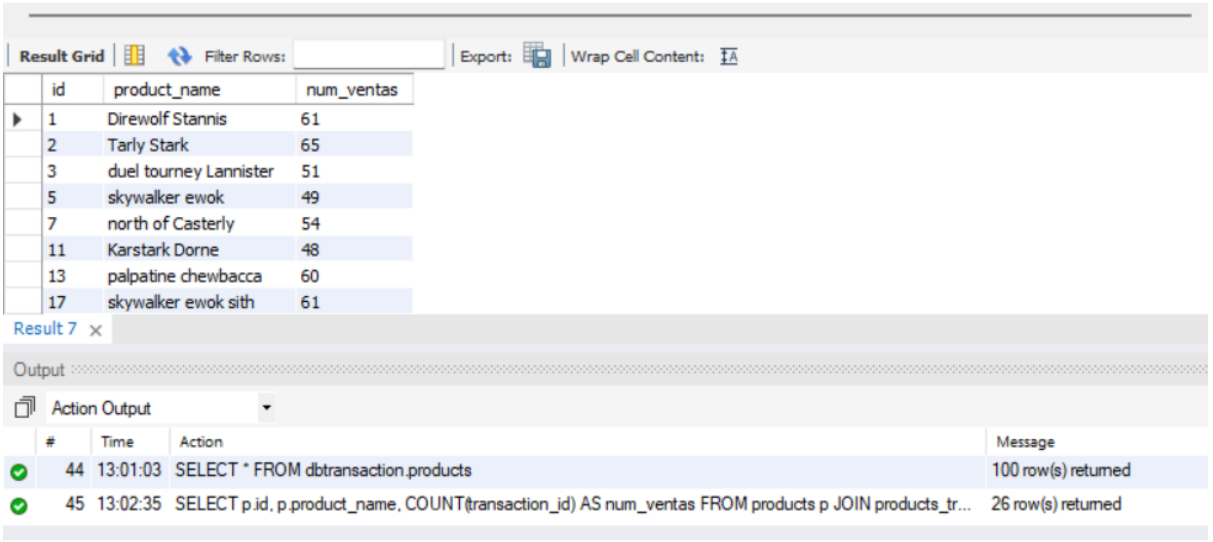
Action Output

#	Time	Action	Message
41	12:57:55	INSERT INTO products_transactions (transaction_id, product_id) SELECT t.id AS transaction_id, p.id AS ...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
42	12:58:42	SELECT * FROM products_transactions	1457 row(s) returned

Ejercicio 1

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

```
224    # Ejercicio 3.1:
225 •  SELECT p.id, p.product_name, COUNT(transaction_id) AS num_ventas
226    FROM products p
227    JOIN products_transactions pt
228    ON p.id = pt.product_id
229    GROUP BY p.id, p.product_name;
```



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

id	product_name	num_ventas
1	Direwolf Stannis	61
2	Tarly Stark	65
3	duel tourney Lannister	51
5	skywalker ewok	49
7	north of Casterly	54
11	Karstark Dorne	48
13	palpatine chewbacca	60
17	skywalker ewok sith	61

Result 7

Output

Action Output

#	Time	Action	Message
44	13:01:03	SELECT * FROM dbtransaction.products	100 row(s) returned
45	13:02:35	SELECT p.id, p.product_name, COUNT(transaction_id) AS num_ventas FROM products p JOIN products_tr...	26 row(s) returned