

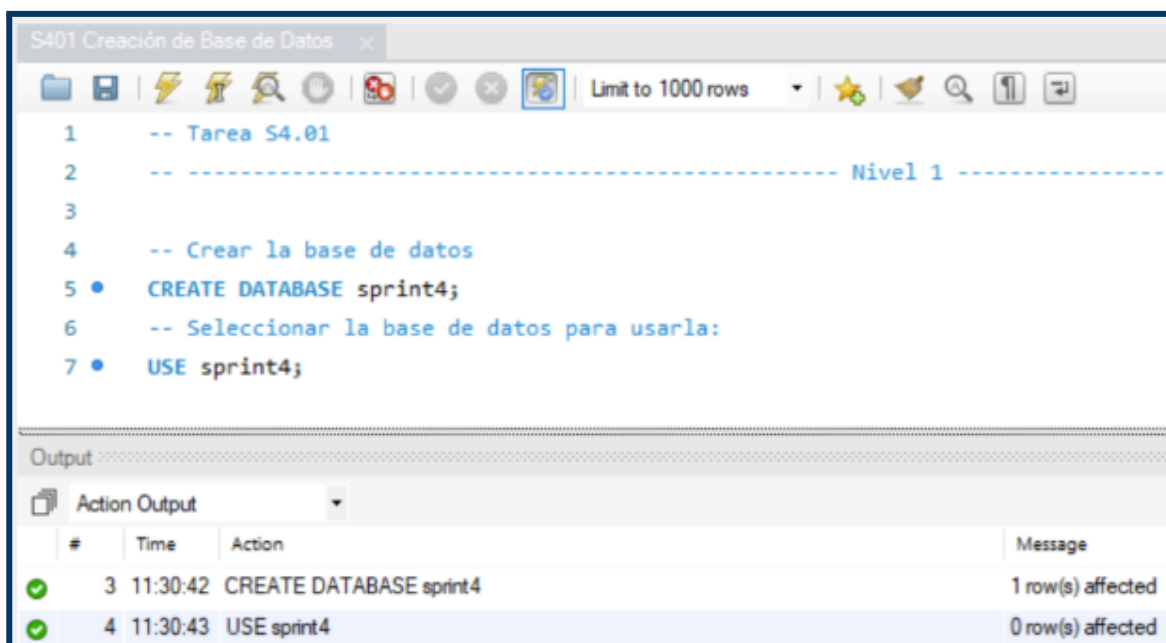
Tarea S4.01. Creación de Base de Datos

1. Nivel 1	1
• Creación de BD	1
• Creación de tablas, relaciones y diagrama ER	1
Concepción de entidades	1
Diseño de relaciones	2
Creación de tablas	3
Modelo Entidad - Relación	7
• Ingesta de data	8
Tabla company	8
Tabla product	8
Tabla user	9
Tabla credit_card	12
Tabla transaction	12
Tabla transaction_product	13
1.1. Ejercicio 1	14
1.2. Ejercicio 2	15
2. Nivel 2	16
2.1. Ejercicio 1	17
3. Nivel 3	18
3.1. Ejercicio 1	18

1. 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:

- Creación de BD



- Creación de tablas, relaciones y diagrama ER

Concepción de entidades

En función de los archivos CSV proporcionados, se determinan **5 entidades** principales:

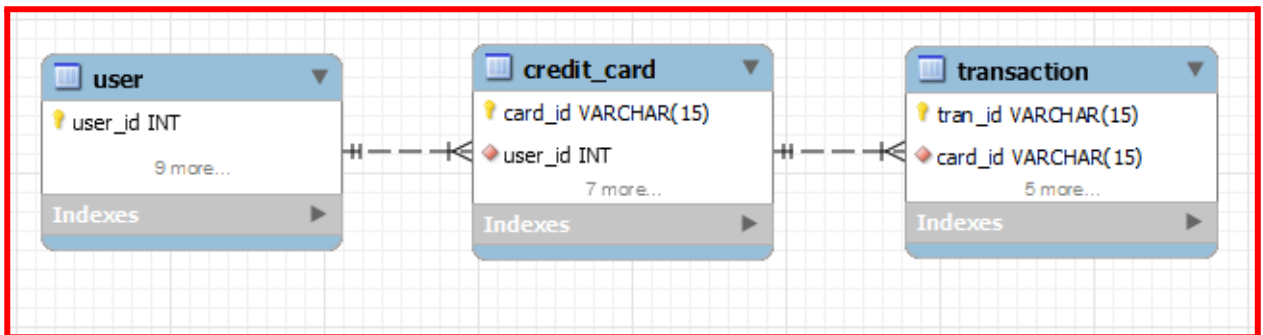
- **Compañía (company)**: Almacena información sobre las empresas registradas.
- **Productos (product)**: Almacena información sobre los productos registrados.
- **Usuarios (user)**: Almacena la información personal de los usuarios.
- **Tarjeta de Crédito (credit_card)**: Almacena información de las tarjetas de crédito, y un vínculo con el usuario al que pertenece.
- **Transacciones (transaction)**: Registra cada transacción incluyendo además de los datos propios, la tarjeta de crédito utilizada, el usuario que realiza la transacción, la compañía vinculada y los productos incluidos.

Diseño de relaciones

En un modelo de datos tradicional, las relaciones entre las entidades siguen un enfoque normalizado, en el que los datos se organizan para minimizar la redundancia y asegurar la integridad referencial. Este tipo de modelo resulta más adecuado cuando el objetivo principal es garantizar la consistencia y eficiencia en el registro de operaciones, como ocurre en bases de datos transaccionales.

Siguiendo este enfoque, la relación entre las entidades usuarios, tarjetas de crédito y transacciones, debería obedecer a una lógica funcional donde:

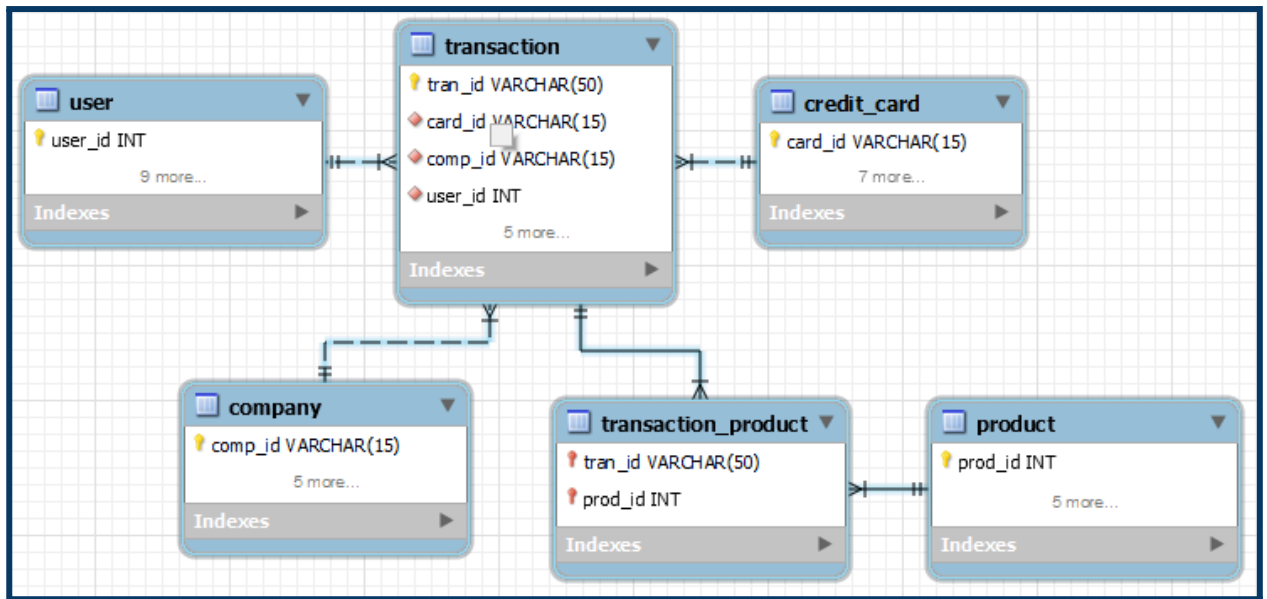
- Cada transacción se realiza con una única tarjeta de crédito.
- Cada tarjeta de crédito pertenece a un único usuario.
- Un usuario puede tener varias tarjetas de crédito.
- Una tarjeta de crédito puede ser utilizada en múltiples transacciones.



Sin embargo, cuando el objetivo es analizar los datos, se recomienda una estructura diferente: el modelo estrella (o su variante, el modelo copo de nieve). Este tipo de modelo está diseñado para optimizar consultas analíticas complejas, típicas de entornos de inteligencia de negocio (BI), donde la prioridad es la agilidad en la consulta y la eficiencia en la agregación de información.

En este contexto, se reorganizan las entidades bajo un enfoque desnormalizado, más adecuado para análisis:

- La tabla **transaction** pasa a ocupar el centro del modelo como tabla de hechos
- Las tablas **user**, **credit_card**, **company** y **product** pasan a ser tablas de dimensiones
- Siguiendo este enfoque, se replantea la relación directa entre **user** y **credit_card**, eliminando dicha dependencia para simplificar la navegación entre dimensiones. Cada dimensión se conecta directamente con la tabla de hechos **transaction**, lo que facilita el análisis por usuario, compañía, producto o tarjeta, sin necesidad de realizar múltiples uniones en cadena durante la ejecución de consultas.
- Por su parte, para gestionar la posibilidad de que una transacción incluya varios productos y, a su vez, cada producto esté vinculado a múltiples transacciones (relación de muchos a muchos), se crea una tabla intermedia llamada **transaction_product**.



Creación de tablas

→ Tabla de Hechos:

★ transaction:

- tran_id (clave primaria)
- tran_timestamp
- tran_amount
- tran_decline
- card_id (clave foránea a la tabla de dimensión credit_card)
- company_id (clave foránea a la tabla de dimensión company)
- user_id (clave foránea a la tabla de dimensión user)

No se tiene en cuenta el listado de productos en la tabla **transaction** para evitar datos redundantes, ya que se define una relación de *muchos a muchos* entre transacciones y productos. Para ello, se crea una tabla intermedia **transaction_product**, que contendrá la información sobre qué productos están involucrados en cada transacción.

→ Tablas de Dimensiones:

★ company:

- comp_id (clave primaria)
- comp_name
- comp_country
- comp_phone
- comp_email
- comp_website

★ user:

- `user_id` (clave primaria)
- `user_name`
- `user_surname`
- `user_email`
- `user_phone`
- `user_address`
- `user_birth_date`
- `user_country`
- `user_city`
- `user_postal_code`

★ `credit_card`:

- `card_id` (clave primaria)
- `card_iban`
- `card_pan`
- `card_pin`
- `card_cvv`
- `card_expiring_date`

No se incluye el `user_id` en la tabla `credit_card` porque, en el modelo estrella propuesto, las tarjetas de crédito y los usuarios no están directamente vinculados.

★ `product`:

- `prod_id` (clave primaria)
- `prod_name`
- `prod_price`
- `prod_colour`
- `prod_weight`
- `prod_warehouse_id`

→ **Tabla de Relación (relación de muchos a muchos entre `transaction` y `product`):**

★ `transaction_product`:

- `tran_id, prod_id` (clave primaria compuesta)
- `tran_id` (clave foránea a la tabla `transaction`)
- `prod_id` (clave foránea a la tabla `product`)

Inicialmente se concibió un campo (con valor 1 por defecto) para almacenar la cantidad de productos del mismo tipo, que se adquieren en una transacción. Sin embargo, debido a que este dato no se proporciona en el CSV, se decidió dejar este campo comentado.

S401 Creación de Base de Dato...

Limit to 1000 rows

```

10  -- Creación de tablas
11  -- Tabla Compañía
12  CREATE TABLE company (
13      comp_id VARCHAR(15) PRIMARY KEY,
14      comp_name VARCHAR(255),
15      comp_country VARCHAR(100),
16      comp_phone VARCHAR(15),
17      comp_email VARCHAR(100),
18      comp_website VARCHAR(100));
19
20  -- Tabla productos
21  CREATE TABLE product (
22      prod_id INT UNSIGNED PRIMARY KEY,
23      prod_name VARCHAR(255),
24      prod_price DECIMAL(10,2),
25      prod_colour VARCHAR(7),
26      prod_weight DECIMAL(3,2),
27      prod_warehouse_id VARCHAR(6));
28
29  -- Tabla user
30  CREATE TABLE user (
31      user_id INT UNSIGNED PRIMARY KEY,
32      user_name VARCHAR(50),
33      user_surname VARCHAR(50),
34      user_phone VARCHAR(15),
35      user_email VARCHAR(100),
36      user_birth_date DATE,
37      user_country VARCHAR(100),
38      user_city VARCHAR(100),
39      user_postal_code VARCHAR(10),
40      user_address VARCHAR(250));
41

```

Output

Action Output

	#	Time	Action	Message
✓	1	16:27:10	CREATE TABLE company (comp_id VARCHAR(15) PRIMARY KEY, comp_name V...	0 row(s) affected
✓	2	16:27:10	CREATE TABLE product (prod_id INT UNSIGNED PRIMARY KEY, prod_name VA...	0 row(s) affected
✓	3	16:27:10	CREATE TABLE user (user_id INT UNSIGNED PRIMARY KEY, user_name VARCH...	0 row(s) affected

S401 Creación de Base de Dato...

Limit to 1000 rows

```

40
41 -- Tabla tarjetas
42 CREATE TABLE credit_card (
43     card_id VARCHAR(15) PRIMARY KEY,
44     card_iban VARCHAR(35) NOT NULL UNIQUE,
45     card_pan VARCHAR(35) NOT NULL UNIQUE,
46     card_pin VARCHAR(4),
47     card_cvv VARCHAR(4),
48     card_track1 VARCHAR(100),
49     card_track2 VARCHAR(100),
50     card_expiring_date DATE /*,
51     /*user_id INT UNSIGNED NOT NULL,
52     CONSTRAINT fk_card_user FOREIGN KEY (user_id) REFERENCES user(user_id) ON DELETE CASCADE*/);
53
54 -- Tabla transacciones
55 CREATE TABLE transaction (
56     tran_id VARCHAR(50) PRIMARY KEY,
57     card_id VARCHAR(15) NOT NULL,
58     comp_id VARCHAR(15) NOT NULL,
59     user_id INT UNSIGNED NOT NULL,
60     tran_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
61     tran_amount DECIMAL(10,2),
62     tran_decline TINYINT,
63     tran_lat FLOAT,
64     tran_longitude FLOAT,
65     CONSTRAINT fk_transaction_company FOREIGN KEY (comp_id) REFERENCES company(comp_id) ON DELETE CASCADE,
66     CONSTRAINT fk_transaction_card FOREIGN KEY (card_id) REFERENCES credit_card(card_id) ON DELETE CASCADE,
67     CONSTRAINT fk_transaction_user FOREIGN KEY (user_id) REFERENCES user(user_id) ON DELETE CASCADE);
68
69 -- Tabla de la relación entre transaction y product
70 CREATE TABLE transaction_product (
71     tran_id VARCHAR(50) NOT NULL,
72     prod_id INT UNSIGNED NOT NULL,
73     -- cantidad SMALLINT UNSIGNED NOT NULL DEFAULT 1,
74     PRIMARY KEY (tran_id, prod_id),
75     CONSTRAINT fk_tp_transaction FOREIGN KEY (tran_id) REFERENCES transaction(tran_id) ON DELETE CASCADE,
76     CONSTRAINT fk_tp_product FOREIGN KEY (prod_id) REFERENCES product(prod_id) ON DELETE CASCADE);

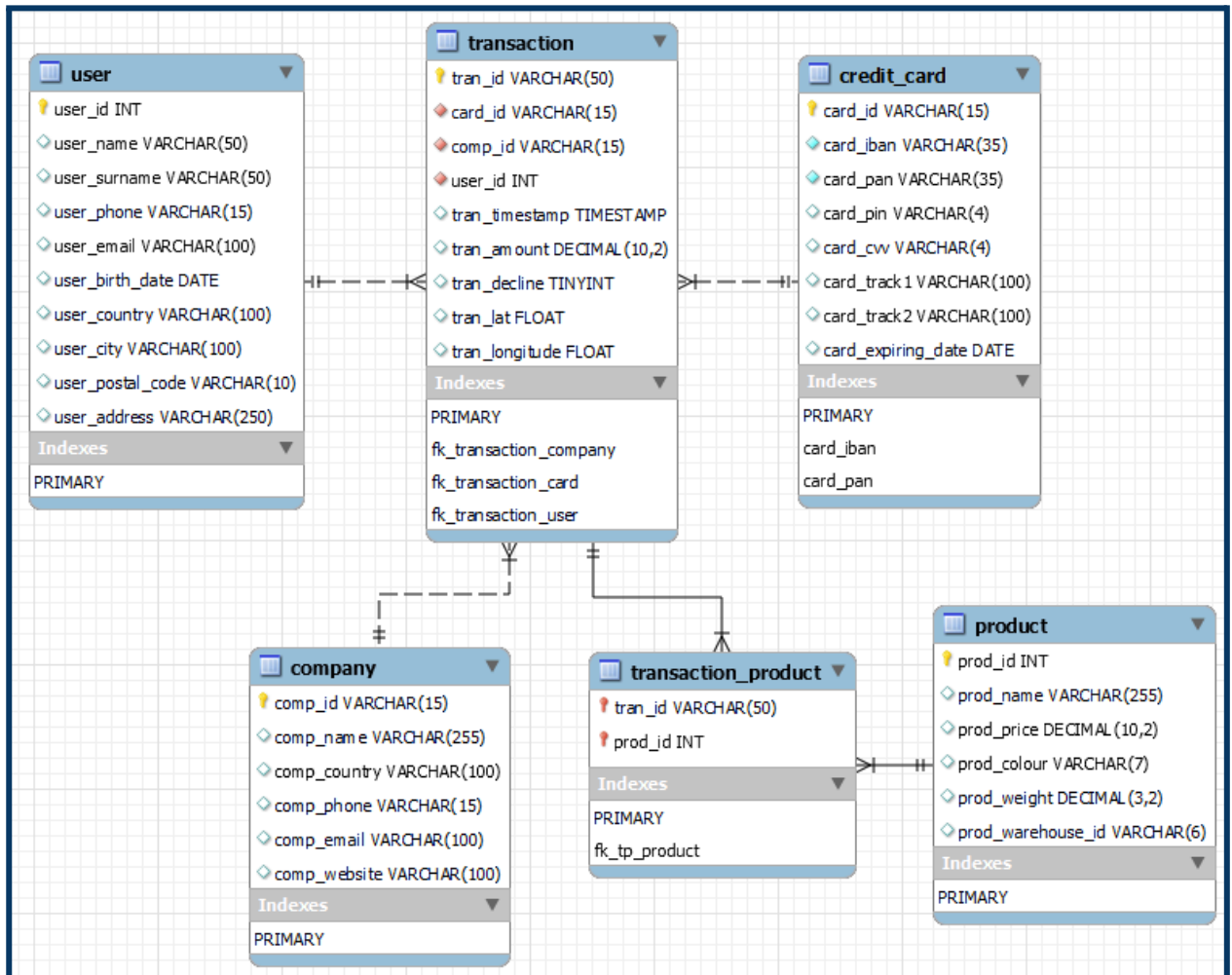
```

Output

Action Output

#	Time	Action	Message
✓ 1	16:27:10	CREATE TABLE company (comp_id VARCHAR(15) PRIMARY KEY, comp_name V...	0 row(s) affected
✓ 2	16:27:10	CREATE TABLE product (prod_id INT UNSIGNED PRIMARY KEY, prod_name VA...	0 row(s) affected
✓ 3	16:27:10	CREATE TABLE user (user_id INT UNSIGNED PRIMARY KEY, user_name VARCH...	0 row(s) affected
✓ 4	16:27:10	CREATE TABLE credit_card (card_id VARCHAR(15) PRIMARY KEY, card_iban VA...	0 row(s) affected
✓ 5	16:27:10	CREATE TABLE transaction (tran_id VARCHAR(50) PRIMARY KEY, card_id VARC...	0 row(s) affected
✓ 6	16:27:10	CREATE TABLE transaction_product (tran_id VARCHAR(50) NOT NULL, prod_id I...	0 row(s) affected

Modelo Entidad - Relación



- Ingesta de data

Tabla company

The screenshot shows a SQL script window titled 'S401 Creación de Base de Datos'. The script contains the following SQL commands:

```

79  -- Ingesta de datos
80  -- Datos de tabla compañía
81  • LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv'
82  INTO TABLE company
83  FIELDS TERMINATED BY ','
84  LINES TERMINATED BY '\\n'
85  IGNORE 1 ROWS
86  (comp_id, comp_name, comp_phone, comp_email, comp_country, comp_website);
87

```

The Output window below shows the execution results:

#	Time	Action	Message
1	16:33:53	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv' INTO...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Tabla product

- El precio del producto es valor decimal, sin embargo, en el CSV el precio incluye un símbolo de moneda (ej: \$10.99). Para corregir esto, se utiliza la función **REPLACE** durante la inserción de los datos, eliminando el símbolo \$ y dejando sólo el valor numérico al almacenar.

Es válido aclarar que se descartó almacenar el tipo de moneda, debido a que actualmente todos los precios que se manejan corresponden a la misma moneda. En caso de que en el futuro se manejen múltiples monedas, se deberá actualizar el modelo para incluir un nuevo campo que la recoja, asegurando la trazabilidad y la correcta interpretación de los precios.

The screenshot shows a SQL script window titled 'S401 Creación de Base de Datos'. The script contains the following SQL commands:

```

88  -- Datos de tabla producto
89  • LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv'
90  INTO TABLE product
91  FIELDS TERMINATED BY ','
92  LINES TERMINATED BY '\\n'
93  IGNORE 1 ROWS
94  (prod_id, prod_name, @price, prod_colour, prod_weight, prod_warehouse_id)
95  SET prod_price = REPLACE(@price, '$', '');
96

```

The Output window below shows the execution results:

#	Time	Action	Message
1	16:35:26	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv' INTO T...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Tabla user

- Los datos se cargan desde tres archivos CSV diferentes: 'users_usa.csv', 'users_uk.csv' y 'users_ca.csv'.
- Se añadió la cláusula **OPTIONALLY ENCLOSED BY ''** para manejar los datos que venían entre comillas.
- La fecha de nacimiento es un campo tipo **DATE**, sin embargo, en el archivo CSV está en formato de texto (ej: "Jan 15, 1990"). Para convertirlo y almacenarlo correctamente en el formato **DATE** de MySQL, se utiliza la función **STR_TO_DATE** durante la inserción de los datos.
- Finalmente también se agregó **'\r\n'** (hasta ahora solo se había trabajado con **\n**) en la cláusula **LINES TERMINATED BY**. Esto debido a que en un primer intento de carga de datos, un warning permitió detectar que no en todos los casos se estaba reconociendo correctamente el final de la línea.

```

95 -- Datos de tabla usuario
96 • LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_usa.csv'
97 INTO TABLE user
98 FIELDS TERMINATED BY ','
99 OPTIONALLY ENCLOSED BY '"'
100 LINES TERMINATED BY '\n'
101 IGNORE 1 ROWS
102 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
103 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
104
105 • LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_uk.csv'
106 INTO TABLE user
107 FIELDS TERMINATED BY ','
108 OPTIONALLY ENCLOSED BY '"'
109 LINES TERMINATED BY '\n'
110 IGNORE 1 ROWS
111 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
112 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
113
114 • LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_ca.csv'
115 INTO TABLE user
116 FIELDS TERMINATED BY ','
117 OPTIONALLY ENCLOSED BY '"'
118 LINES TERMINATED BY '\n'
119 IGNORE 1 ROWS
120 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
121 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
122

```

#	Time	Action	Message
1	16:09:51	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Serv...	113 row(s) affected, 37 warning(s): 1262 Row 10 was truncated; it contained more data than there were input columns 1262 Row
2	16:09:51	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Serv...	35 row(s) affected, 15 warning(s): 1262 Row 8 was truncated; it contained more data than there were input columns 1262 Row 12
3	16:09:51	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Serv...	58 row(s) affected, 17 warning(s): 1262 Row 4 was truncated; it contained more data than there were input columns 1262 Row 12

Los warning hacían referencia a datos truncados durante la ingesta de datos, debido a que algunos valores excedían el tamaño permitido para los campos correspondientes. Estos warnings venían acompañados de la referencia a las filas específicas donde se producía este error. Por lo tanto, se buscó una de estas filas problemáticas para investigar el motivo del truncamiento y asegurarse de que los datos se ingresaron correctamente, ajustando el tamaño de los campos según fuera necesario.

S401 Creación de Base de Datos

```
-- Mostrar datos de tabla user  
select *  
FROM user;
```

Result Grid

	user_id	user_name	user_surname	user_phone	user_email	user_birth_date	user_country	user_city	user_postal_code	user_address
1	Zeus	Gamble		1-282-581-0551	interdum.enim@protonmail.edu	1985-11-17	United States	Lovell	73544	348-7818 Sagittis St.
2	Garrett	Mccormell	(718)	257-2412	integer.vitae.nibh@protonmail.org	1992-08-23	United States	Des Moines	59464	903 Sit Ave
3	Ciaran	Harrison	(522)	598-1365	interdum.feugiat@aol.org	1998-04-29	United States	Columbus	56518	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-	3269	ornare.egestas@icloud.eu	1989-02-18	United States	Kailua	77417	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce	1-554-541-	2077	et.malesuada.fames@hotmail.org	1998-09-26	United States	Sandy	31564	341-2821 Ultrices Av.
6	Joel	Tyson	(718)	288-8020	gravida.nunc.sed@yahoo.ca	1989-10-15	United States	Nashville	96838	888-2799 Amet Street
7	Rafael	Jimenez	(817)	689-0478	eget@outlook.ca	1981-12-04	United States	Hillsboro	29874	8627 Malesuada Rd.
8	Nissim	Franks	(692)	157-3469	egestas.aliquam.fringilla@google.ca	1993-08-01	United States	Jackson	61750	Ap #251-7144 Integer St.
9	Mannix	Mcdain	(590)	883-2184	aliquam.nisl@outlook.com	1987-01-24	United States	Richmond	35987	647-3080 Lacus. St.
10	Robert	Mccarthy	(324)	746-6771	fermentum@protonmail.com	1984-04-30	United States	Eugene	85526	P.O. Box 773, 3594 Ornare St.* 11, Joan,Baird,...
12	Benedict	Wheeler	1-515-824-	2855	tincidunt.donec.vitae@hotmail.couk	1999-08-06	United States	Lewiston	92393	748-8694 Porttitor Avenue
13	Allegra	Stanton	1-927-753-	6488	proin.eget@protonmail.ca	1990-05-19	United States	Kearney	14947	4457 Ante. Av.
14	Sara	Flynn	1-311-646-	9333	integer@outlook.net	1988-12-27	United States	Warren	20288	P.O. Box 865, 4397 Ante St.* 15,Noelani,Patric...
16	Eric	Roth	1-218-549-	8253	lorem.sit@yahoo.net	1988-09-07	United States	Reading	96697	P.O. Box 541, 5137 Non Road* 17,Bruce,Gill,(7...
18	Russell	Jimenez	(657)	779-2438	ord@outlook.edu	1993-08-26	United States	Hattiesburg	75647	4095 Quam Rd.

Output

Action Output

#	Time	Action	Message
1	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	113 row(s) affected, 37 warning(s): 1262 Row 10 was truncated; it contained more data than there were input columns 1262 Row 13 was trun
2	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	35 row(s) affected, 15 warning(s): 1262 Row 8 was truncated; it contained more data than there were input columns 1262 Row 12 was trunc
3	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	58 row(s) affected, 17 warning(s): 1262 Row 4 was truncated; it contained more data than there were input columns 1262 Row 12 was trunc
4	16:12:29	select * FROM user LIMIT 0, 5000	206 row(s) returned

En este punto, se pudo comprobar que en el `user_address` (el último de los campos), se estaba almacenando junto con el valor correspondiente, los datos pertenecientes a la siguiente fila. Esto indicaba que no se estaba detectando correctamente el salto de línea al procesar los datos, lo que provocaba la mezcla de registros en una sola fila.

Este tipo de problema puede ocurrir cuando no se gestionan adecuadamente los delimitadores de fila, lo que impide una separación correcta de los datos. Es importante tener en cuenta que el delimitador de línea puede variar según el sistema operativo donde se creó el archivo. Por ejemplo, en sistemas Windows, el salto de línea suele estar representado por `\r\n`, mientras que en sistemas Unix/Linux se usa solo `\n`.

En este contexto se decidió:

- Vaciar la tabla
- Agregar ' \r\n' en la cláusula **LINES TERMINATED BY**.
- Volver a cargar los datos

Vaciar la tabla

The screenshot shows a SQL Server Enterprise Manager window titled "S401 Creación de Base de Datos...". The query window contains the following SQL code:

```
128 -- Vaciar la tabla
129 TRUNCATE TABLE user;
130
```

The Output pane shows the execution results:

#	Time	Action	Message
1	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	113 row(s) affected, 37
2	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	35 row(s) affected, 15
3	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	58 row(s) affected, 17
4	16:12:29	select * FROM user LIMIT 0, 50000	206 row(s) returned
5	16:14:47	TRUNCATE TABLE user	0 row(s) affected

Modificación y nueva ingesta de datos

The screenshot shows a SQL Server Enterprise Manager window titled "S401 Creación de Base de Datos...". The query window contains the following SQL code:

```
95 -- Datos de tabla usuario
96 LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\users_usa.csv'
97 INTO TABLE user
98 FIELDS TERMINATED BY ','
99 OPTIONALLY ENCLOSED BY '"'
100 LINES TERMINATED BY '\r\n'
101 IGNORE 1 ROWS
102 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
103 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
104
105 LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\users_uk.csv'
106 INTO TABLE user
107 FIELDS TERMINATED BY ','
108 OPTIONALLY ENCLOSED BY '"'
109 LINES TERMINATED BY '\r\n'
110 IGNORE 1 ROWS
111 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
112 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
113
114 LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\users_ca.csv'
115 INTO TABLE user
116 FIELDS TERMINATED BY ','
117 OPTIONALLY ENCLOSED BY '"'
118 LINES TERMINATED BY '\r\n'
119 IGNORE 1 ROWS
120 (user_id, user_name, user_surname, user_phone, user_email, @user_birth_date, user_country, user_city, user_postal_code, user_address)
121 SET user_birth_date = STR_TO_DATE(@user_birth_date, '%b %d, %Y');
```

The Output pane shows the execution results:

#	Time	Action	Message
1	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	113 row(s) affected, 37 warning(s): 1262 Row 10 was truncated; it contained more data than there were input colum
2	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	35 row(s) affected, 15 warning(s): 1262 Row 8 was truncated; it contained more data than there were input columns
3	16:09:51	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	58 row(s) affected, 17 warning(s): 1262 Row 4 was truncated; it contained more data than there were input columns
4	16:12:29	select * FROM user LIMIT 0, 50000	206 row(s) returned
5	16:14:47	TRUNCATE TABLE user	0 row(s) affected
6	16:18:52	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0
7	16:18:52	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0
8	16:18:52	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Serv...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0

Tabla `credit_card`

- La fecha de expiración de la tarjeta de crédito es un campo tipo DATE, sin embargo, en el archivo CSV está en un formato distinto al admitido por MySQL (ej: "01/15/25"). Para almacenarla correctamente se utiliza la función `STR_TO_DATE` durante la inserción de los datos.

The screenshot shows a SQL script window titled "S401 Creación de Base de Datos" with the following SQL code:

```

126 -- Datos de tabla tarjeta de credito
127 LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\credit_cards.csv'
128 INTO TABLE credit_card
129 FIELDS TERMINATED BY ','
130 LINES TERMINATED BY '\n'
131 IGNORE 1 ROWS
132 (card_id, @ignorar, card_iban, card_pan, card_pin, card_cvv, card_track1, card_track2, @card_expiring_date)
133 SET card_expiring_date = STR_TO_DATE(@card_expiring_date, '%m/%d/%y');
134

```

The Output window below shows the execution results:

#	Time	Action	Message
1	16:37:44	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\credit_cards.csv' INT...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0

Tabla `transaction`

- Se define el delimitador como `;`, que es el utilizado en este archivo CSV específico.
- Se omiten la columna `prod_id`, ya que esta información será almacenada en la tabla de relación entre transacciones y productos.

The screenshot shows a SQL script window titled "S401 Creación de Base de Datos" with the following SQL code:

```

133
134 -- Datos de tabla transaccion
135 LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv'
136 INTO TABLE transaction
137 FIELDS TERMINATED BY ';'
138 LINES TERMINATED BY '\n'
139 IGNORE 1 ROWS
140 (tran_id, card_id, comp_id, tran_timestamp, tran_amount, tran_decline, @ignorar, user_id, tran_lat, tran_longitude);
141

```

The Output window below shows the execution results:

#	Time	Action	Message
1	16:38:45	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv' INT...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0

Tabla `transaction_product`

→ En este caso, tenemos una situación en la que la información de productos asociados a transacciones está almacenada en un solo campo dentro de un archivo CSV, con los valores de los productos separados por comas. Dado que MySQL no tiene una función nativa para separar directamente los valores de un campo de texto, se propone la siguiente solución:

1. Creación de la Tabla Temporal

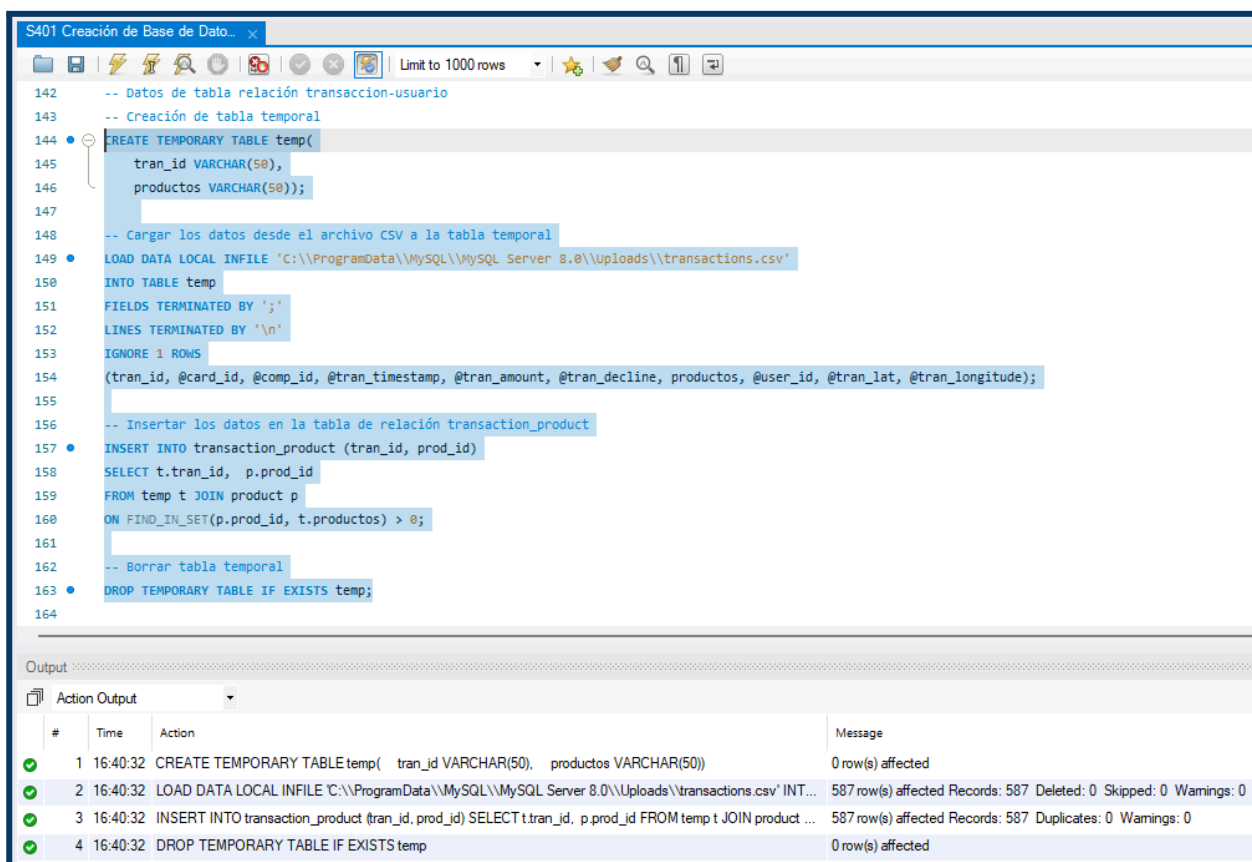
Se crea una tabla temporal para almacenar de cada transacción, su ID y una cadena de texto con los identificadores de los productos, separados por comas.

2. Carga de los Datos desde el Archivo CSV a la Tabla Temporal

3. Inserción de los registros en la tabla de relación `transaction_product` con la función `FIND_IN_SET`

Para cada transacción (`tran_id`), se verifica producto a producto (`prod_id`) de la tabla `product`, si está contenido en la cadena de productos almacenada en la tabla temporal. Se insertan en la tabla `transaction_product` únicamente los pares (`tran_id`, `prod_id`) de los `prod_id` que se encuentran.

4. Eliminación de la tabla temporal



```

142 -- Datos de tabla relación transaction_usuario
143 -- Creación de tabla temporal
144 CREATE TEMPORARY TABLE temp(
145     tran_id VARCHAR(50),
146     productos VARCHAR(50));
147
148 -- Cargar los datos desde el archivo CSV a la tabla temporal
149 LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv'
150 INTO TABLE temp
151 FIELDS TERMINATED BY ','
152 LINES TERMINATED BY '\n'
153 IGNORE 1 ROWS
154 (tran_id, @card_id, @comp_id, @tran_timestamp, @tran_amount, @tran_decline, productos, @user_id, @tran_lat, @tran_longitude);
155
156 -- Insertar los datos en la tabla de relación transaction_product
157 INSERT INTO transaction_product (tran_id, prod_id)
158 SELECT t.tran_id, p.prod_id
159 FROM temp t JOIN product p
160 ON FIND_IN_SET(p.prod_id, t.productos) > 0;
161
162 -- Borrar tabla temporal
163 DROP TEMPORARY TABLE IF EXISTS temp;
164

```

Output:

#	Time	Action	Message
1	16:40:32	CREATE TEMPORARY TABLE temp(tran_id VARCHAR(50), productos VARCHAR(50))	0 row(s) affected
2	16:40:32	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv' INT...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0
3	16:40:32	INSERT INTO transaction_product (tran_id, prod_id) SELECT t.tran_id, p.prod_id FROM temp t JOIN product ...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
4	16:40:32	DROP TEMPORARY TABLE IF EXISTS temp	0 row(s) affected

Es importante señalar que esta consulta resulta costosa en términos de rendimiento, ya que recorre todas las transacciones registradas y, por cada una de ellas, compara todos los productos existentes con la cadena de productos almacenada. Esto puede hacer que el proceso sea considerablemente lento si se manejan grandes volúmenes de datos.

Se optó por esta solución siguiendo las indicaciones acotadas del ejercicio docente; sin embargo, ante una situación similar en un entorno productivo, es recomendable optar por alternativas más simples mediante procesos ETL externos al motor de base de datos.

1.1. Ejercicio 1

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

Explicación:

- Se realiza una subconsulta sobre la tabla `transaction` para obtener un listado de los usuarios (`user_id`) que tienen más de 30 transacciones.
- En la consulta principal sobre la tabla `user` se recogen los datos de los usuarios (`user_id`, `user_name` y `user_surname`) cuyos id aparecen en el listado anterior.

The screenshot shows a database management tool interface. The top pane displays a SQL query for 'Ejercicio 1' which uses a subquery to find users with more than 30 transactions. The bottom pane shows the 'Result Grid' with 4 rows of user data. Below the grid, the 'Output' section shows the execution of the query, indicating that 4 rows were returned.

```

157 -- Ejercicio 1 --
158 -- Usuarios con más de 30 transacciones (con subconsulta y utilizando al menos 2 tablas)
159 SELECT u.user_id, u.user_name, u.user_surname
160 FROM user u
161 WHERE u.user_id IN (
162     SELECT t.user_id
163     FROM transaction t
164     GROUP BY t.user_id
165     HAVING COUNT(t.tran_id) > 30);
  
```

user_id	user_name	user_surname
92	Lynn	Riddle
267	Ocean	Nelson
272	Hedwig	Gilbert
275	Kenyon	Hartman
NULL	NULL	NULL

user 1 x

Output

Action Output

#	Time	Action	Message
1	12:10:09	SELECT u.user_id, u.user_name, u.user_surname FROM user u WHERE u.user_id IN (SE...	4 row(s) returned

1.2. 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.

Explicación:

- A partir del conjunto de datos obtenido mediante la intersección de las tablas `transaction`, `credit_card` y `company`, se filtran los registros correspondientes a la compañía 'Donec Ltd'.
- Luego, los datos se agrupan por IBAN de la tarjeta de crédito y se calcula la media del importe (`tran_amount`) de las transacciones asociadas a cada tarjeta.

The screenshot shows a database management tool interface. The top pane displays a SQL query for 'Ejercicio 2' which calculates the average transaction amount by credit card IBAN for the company 'Donec Ltd'. The query uses a join between the 'transaction', 'credit_card', and 'company' tables. The bottom pane shows the 'Result Grid' with one row of data: IBAN 'PT87806228135092429456346' and a media value of 203.72. Below the grid, the 'Output' section shows a message indicating that 1 row(s) were returned.

```

172 -- Ejercicio 2 --
173 -- Media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd (utiliza por lo menos 2 tablas)
174 • SELECT cc.card_iban AS IBAN, ROUND(AVG(t.tran_amount), 2) AS media
175 FROM transaction t
176 JOIN credit_card cc ON t.card_id = cc.card_id
177 JOIN company c ON t.comp_id = c.comp_id
178 WHERE c.comp_name = 'Donec Ltd'
179 GROUP BY cc.card_iban;
180

```

IBAN	media
PT87806228135092429456346	203.72

Result 2 x

Output

Action Output

#	Time	Action	Message
1	14:17:16	SELECT cc.card_iban AS IBAN, ROUND(AVG(t.tran_amount), 2) AS media FROM transaction t JOIN credit_card cc ON t.card_id = ...	1 row(s) returned

2. 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:

Explicación del llenado de la tabla:

- Mediante la función `ROW_NUMBER()` se enumeran las transacciones por tarjeta de crédito, ordenadas por `tran_timestamp` de forma descendente.
- La elección de `ROW_NUMBER()` sobre `RANK()` se debe a que la elegida asegura una numeración única para las transacciones, aunque se diera el caso de más de una transacción con el mismo `timestamp`.
- Se filtran sólo las últimas 3 transacciones (`row_num >= 3`) por cada tarjeta.
- Se determina el estado de la tarjeta. Si exactamente las tres transacciones filtradas han sido rechazadas (`SUM(tran_decline)=3`), el estado de la tarjeta será `0` (inactiva). Si no, será `1` (activa).

The screenshot shows a database management tool interface with the following components:

- SQL Editor:** Contains SQL code for creating a table and inserting data.


```

182 -- Nivel 2
183 -- Crear una nueva tabla de estado de tarjeta
184 CREATE TABLE card_status (
185     card_id VARCHAR(15) PRIMARY KEY,
186     -- updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
187     status BOOLEAN);
188
189 -- Llenando la tabla de estado, definiendolo en función de las últimas 3 transacciones
190 INSERT INTO card_status (card_id, status)
191 SELECT aux.card_id, IF(SUM(aux.tran_decline) = 3, 0, 1) AS status
192 FROM (SELECT t.card_id, t.tran_decline, ROW_NUMBER() OVER (PARTITION BY t.card_id ORDER BY t.tran_timestamp DESC) AS row_num
193      FROM transaction t) aux
194 WHERE aux.row_num <= 3
195 GROUP BY aux.card_id;
196
197 -- Mostrando los registros de la tabla de estatus
198 SELECT * FROM card_status;
      
```
- Result Grid:** Displays the contents of the `card_status` table.

card_id	status
CcU-2938	1
CcU-2945	1
CcU-2952	1
CcU-2959	1
CcU-2966	1
CcU-2973	1
CcU-2980	1
CcU-2987	1
CcU-2994	1
CcU-3001	1
CcU-3008	1
CcU-3015	1
CcU-3022	1
- Output:** Shows the execution log with three actions:

#	Time	Action	Message
1	11:10:49	CREATE TABLE card_status (card_id VARCHAR(15) PRIMARY KEY, -- updated TIMESTAMP DEFAULT CURRENT_...	0 row(s) affected
2	11:10:49	INSERT INTO card_status (card_id, status) SELECT aux.card_id, IF(SUM(aux.tran_decline) = 3, 0, 1) AS status FROM (SEL...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
3	11:10:49	SELECT * FROM card_status LIMIT 0, 1000	275 row(s) returned

2.1. Ejercicio 1

¿Cuántas tarjetas están activas?

Hay activas 275 tarjetas

The screenshot shows a SQL IDE window titled "S401 Creación de Base de Dato...". The query editor contains the following SQL code:

```
-- Ejercicio 1
-- Consultando cuántas tarjetas están activas
select SUM(status)
FROM card_status;
```

The "Result Grid" shows a single row with the value 275 for the column SUM(status).

The "Output" pane shows the "Action Output" table with the following rows:

#	Time	Action	Message
1	11:10:49	CREATE TABLE card_status (card_id VARCHAR(15) PRIMARY KEY, -- updated TIMESTAMP DEFAULT CURRENT_...	0 row(s) affected
2	11:10:49	INSERT INTO card_status (card_id, status) SELECT aux.card_id, IF(SUM(aux.tran_decline) = 3, 0, 1) AS status FROM (SEL...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
3	11:10:49	SELECT * FROM card_status LIMIT 0, 1000	275 row(s) returned
4	11:10:49	select SUM(status) FROM card_status LIMIT 0, 1000	1 row(s) returned

Hay registradas 275 tarjetas

The screenshot shows a SQL IDE window titled "S401 Creación de Base de Dato...". The query editor contains the following SQL code:

```
-- Cantidad de tarjetas registradas
SELECT COUNT(card_id)
FROM credit_card;
```

The "Result Grid" shows a single row with the value 275 for the column COUNT(card_id).

The "Output" pane shows the "Action Output" table with the following rows:

#	Time	Action	Message
1	14:52:51	SELECT COUNT(card_id) FROM credit_card LIMIT 0, 50000	1 row(s) returned

Todas las tarjetas están activas.

3. Nivel 3

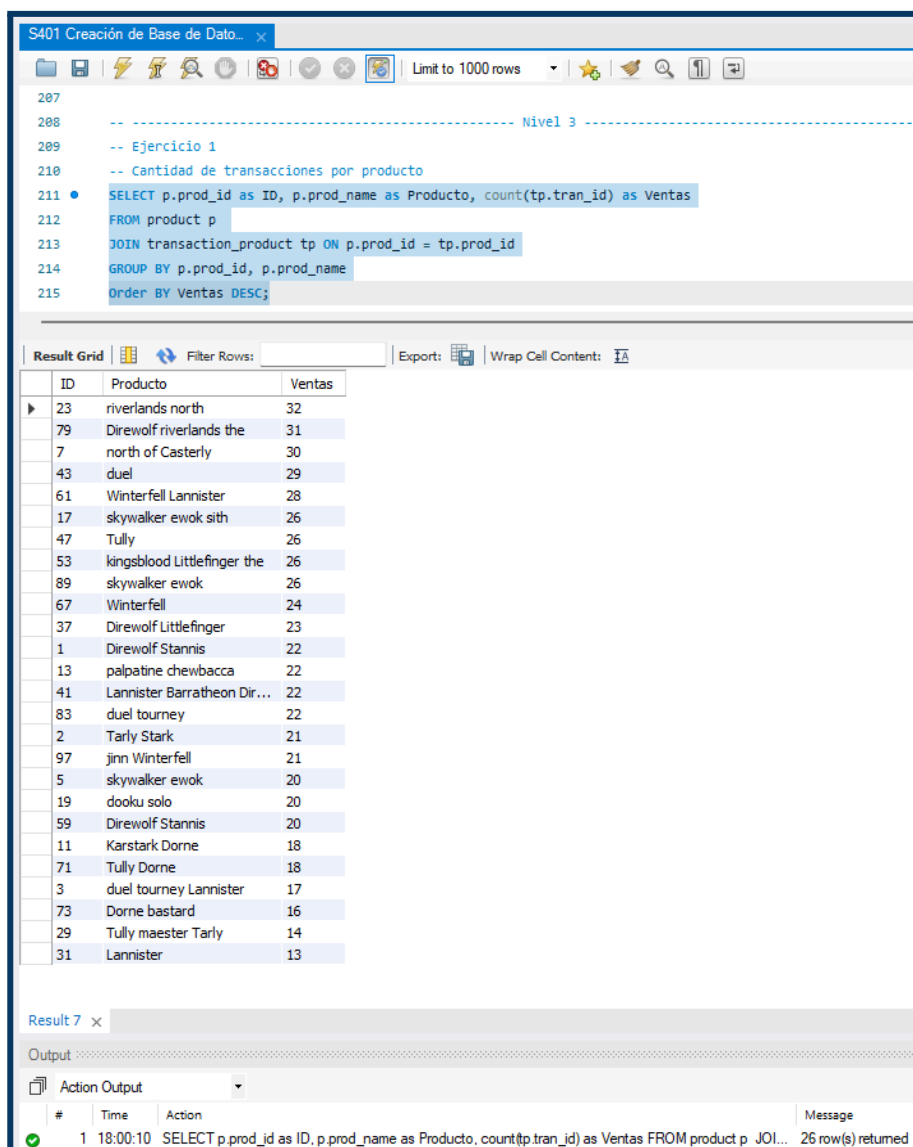
Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes product_ids. Genera la siguiente consulta:

Es la tabla `transaction_product` creada desde la concepción del modelo relacional, como tabla intermedia para gestionar la relación de muchos a muchos entre las entidades `product` y `transaction`. Es una solución estándar en bases para manejar relaciones de este tipo.

3.1. Ejercicio 1

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

- Como no se está recogiendo la cantidad de productos de un mismo tipo dentro de una transacción, la cantidad de ventas se interpreta como el número de transacciones en las que cada producto ha sido incluido.



The screenshot shows a SQL IDE window titled "S401 Creación de Base de Datos...". The query editor contains the following SQL code:

```

207
208  ----- Nivel 3 -----
209  -- Ejercicio 1
210  -- Cantidad de transacciones por producto
211  SELECT p.prod_id as ID, p.prod_name as Producto, count(tp.tran_id) as Ventas
212  FROM product p
213  JOIN transaction_product tp ON p.prod_id = tp.prod_id
214  GROUP BY p.prod_id, p.prod_name
215  Order BY Ventas DESC;
  
```

Below the query editor, the "Result Grid" shows the results of the query. The table has three columns: ID, Producto, and Ventas. The results are sorted by Ventas in descending order.

ID	Producto	Ventas
23	riverlands north	32
79	Direwolf riverlands the	31
7	north of Casterly	30
43	duel	29
61	Winterfell Lannister	28
17	skywalker ewok sith	26
47	Tully	26
53	kingsblood Littlefinger the	26
89	skywalker ewok	26
67	Winterfell	24
37	Direwolf Littlefinger	23
1	Direwolf Stannis	22
13	palpatine chewbacca	22
41	Lannister Barratheon Dir...	22
83	duel tourney	22
2	Tarly Stark	21
97	jinn Winterfell	21
5	skywalker ewok	20
19	dooku solo	20
59	Direwolf Stannis	20
11	Karstark Dorne	18
71	Tully Dorne	18
3	duel tourney Lannister	17
73	Dorne bastard	16
29	Tully maester Tarly	14
31	Lannister	13

At the bottom, the "Output" section shows the execution details:

```

Result 7 x
Output
Action Output
# Time Action Message
1 18:00:10 SELECT p.prod_id as ID, p.prod_name as Producto, count(tp.tran_id) as Ventas FROM product p JOIN... 26 row(s) returned
  
```