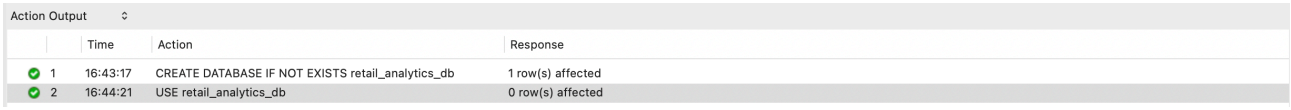
DESCARGA LOS ARCHIVOS CSV, ESTUDIALES Y DISEÑA UNA BASE DE DATOS CON UN ESQUEMA DE ESTRELLA QUE CONTENGA, AL MENOS 4 TABLAS

Para crear la base de datos, utilizaremos una consulta:

```
CREATE DATABASE IF NOT EXISTS retail_analytics_db;
USE retail_analytics_db;
```

Resultado de la ejecución de la consulta:



Action Output	Time	Action	Response
1	16:43:17	CREATE DATABASE IF NOT EXISTS retail_analytics_db	1 row(s) affected
2	16:44:21	USE retail_analytics_db	0 row(s) affected

Antes de empezar a importar archivos csv en la base de datos, familiaricémonos con su contenido, determinemos los futuros tipos de columnas que serán adecuados para nuestra base de datos y creemos tablas.

Company

El documento Companies.csv es una tabla con los datos de las empresas con las que se realizan transacciones. Contiene información como

- Company ID,
- Company name,
- Phone,
- Email,
- Country,
- Website.

Todos los columnas son de tipo de datos VARCHAR con diferentes longitudes.

La tabla contiene una descripción del campo business_id de la tabla Transactions, y por lo tanto es una tabla de tipo dimensión en nuestra base de datos.

El identificador único de la empresa es su Company_ID, lo que significa que en nuestra futura tabla - será el PK de la tabla de empresas.

Por lo tanto, para crear la tabla Empresa utilizamos una consulta:

```
CREATE TABLE IF NOT EXISTS company(
	Company_ID VARCHAR(36)PRIMARY KEY,
	Company_name VARCHAR(150),
	Phone VARCHAR(20),
	Email VARCHAR(150),
	Country VARCHAR(100),
	Website VARCHAR(255));
```

Resultado de la ejecución de la consulta:

Action Output	Time	Action	Response
1	16:47:06	CREATE TABLE IF NOT EXISTS company(Company_ID VAR...	0 row(s) affected

He utilizado la siguiente consulta para importar datos a la tabla de empresas:

```
65 •  LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/downloads/companies.csv'
66    INTO TABLE company
67    FIELDS TERMINATED BY ','
68    ENCLOSED BY '"'
69    LINES TERMINATED BY '\n'
70    IGNORE 1 ROWS;
71
```

100% 1:61

Action Output

	Time	Action	Response
1	19:27:33	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Vamos a ejecutar una consulta SELECT para asegurarnos de que los datos se cargan correctamente:

```
72 •  SELECT * FROM company;
73
```

100% 25:69

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

Company_ID	Company_name	Phone	Email	Country	Website
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://instagram.com/site
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/group/9
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/sub/cars

company 56

Action Output

	Time	Action	Response
1	19:27:33	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
2	19:28:48	SELECT * FROM company	100 row(s) returned

Como resultado de la consulta, podemos ver que cada valor coincide con la finalidad del campo y está representado por el valor correcto.

Credit cards

El documento Credit_cards.csv es información sobre las tarjetas de crédito que se utilizaron para todas las transacciones en nuestro sistema. Contiene información como

- ID,
- User_id,
- IBAN,
- PAN,
- PIN,
- CVV,
- Track_1,
- Track_2,
- Expiring date.

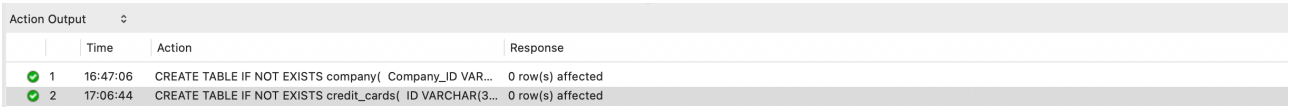
Puede encontrar una descripción detallada de cada tipo para el campo de datos actual aquí.

Los nuevos campos Track_1 y Track_2, que contienen datos de pistas de tarjetas magnéticas deben almacenarse de forma encriptada. Para nuestra base de datos es adecuado el tipo VARCHAR con limitación de longitud de cadena según el valor máximo de cadena especificado en ISO para estos datos.

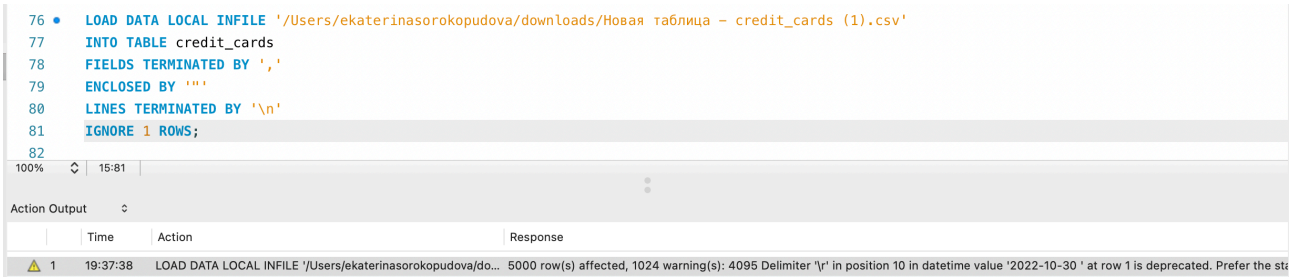
La tabla contiene una descripción del campo card_id de la tabla de transacciones, y por lo tanto es una tabla de tipo dimensión.

```
CREATE TABLE IF NOT EXISTS credit_cards(
        ID VARCHAR(36)PRIMARY KEY,
        User_id VARCHAR(36),
        IBAN VARCHAR (34),
        PAN VARCHAR (34),
        PIN CHAR (4),
        CVV CHAR (4),
        Track_1 VARCHAR(79),
        Track_2 VARCHAR(40),
        Expiring_date DATE);
```

Resultado de la ejecución de la consulta:

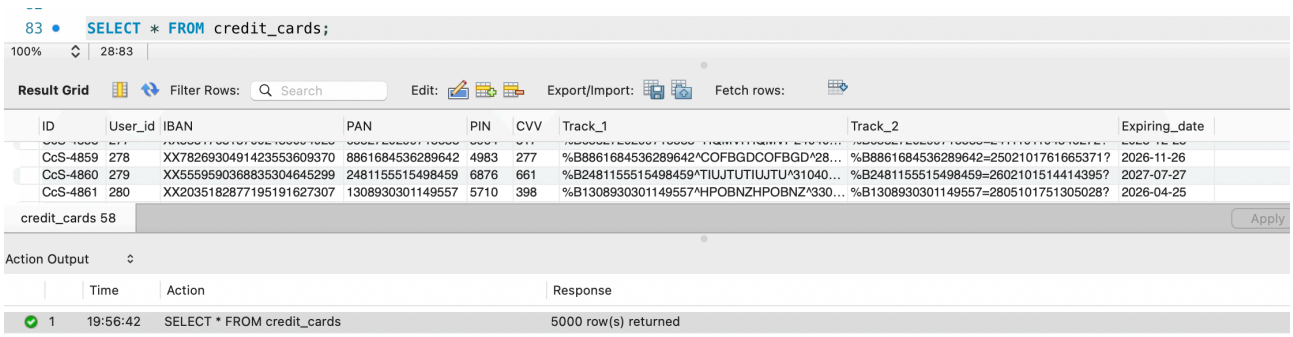


Utilicé la siguiente consulta para importar los datos en la tabla de tarjetas de crédito, preprocesando el archivo en excel para cambiar el formato de los datos a aaaa-mm-dd y asegurándome de que no afectaba a la calidad de los valores presentados.



La advertencia con el texto "Delimitador “\r” en la posición 10 en el valor datetime “2022-10-30 ” en la fila 1 está obsoleto. Prefiera el estándar". informa de que se detecta un carácter de carro especial al final de algunas líneas del documento, pero actualmente mysql puede cargar datos con esta advertencia sin errores. En el futuro, vale la pena eliminar el desplazamiento de carro con un editor de texto de antemano.

Vamos a ejecutar una consulta SELECT para asegurarnos de que los datos se cargan correctamente:



Como resultado de la consulta, podemos ver que cada valor coincide con la finalidad del campo y está representado por el valor correcto.

Users

Los documentos european_users.csv y american_users.csv son absolutamente idénticos en el conjunto de campos que contienen. Describen a los usuarios que han realizado transacciones en nuestro sistema y contienen información sobre:

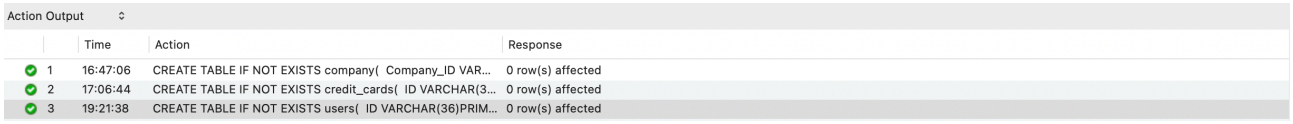
- ID,
- Nombre,
- Apellidos,
- Teléfono,
- Correo electrónico,
- Fecha nacimiento,
- País,
- Ciudad,
- Código postal,
- Dirección.

Todos los campos excepto el campo Birth_date son campos con el tipo de datos VARCHAR.

Dado que ambas tablas contienen información común, utilizaremos una tabla llamada Users. Los campos de la tabla son descripciones del campo User_id de la tabla de transacciones, por lo que la tabla de usuarios es una tabla de dimensión en la base de datos.

```
CREATE TABLE IF NOT EXISTS users(
        ID VARCHAR(36)PRIMARY KEY,
        Name VARCHAR(100),
        Surname VARCHAR(100),
        Phone VARCHAR(20),
        Email VARCHAR(150),
        Birth_date DATE,
        Country VARCHAR(100),
        City VARCHAR(100),
        Postal_code VARCHAR(10),
        Address VARCHAR(255));
```

Resultado de la ejecución de la consulta:



Action Output

	Time	Action	Response
1	16:47:06	CREATE TABLE IF NOT EXISTS company(Company_ID VAR...	0 row(s) affected
2	17:06:44	CREATE TABLE IF NOT EXISTS credit_cards(ID VARCHAR(3...	0 row(s) affected
3	19:21:38	CREATE TABLE IF NOT EXISTS users(ID VARCHAR(36)PRIM...	0 row(s) affected

Antes de ejecutar la siguiente consulta, al igual que en el caso de las tarjetas de crédito, he preferido cambiar el tipo de fecha a aaaa-mm-dd para trabajar correctamente con campos de fecha en el futuro, y habiéndome familiarizado previamente con el fichero csv, he corregido los delimitadores de datos en la consulta por deterioro a ‘;’. Así que utilicé consultas para importar los datos:

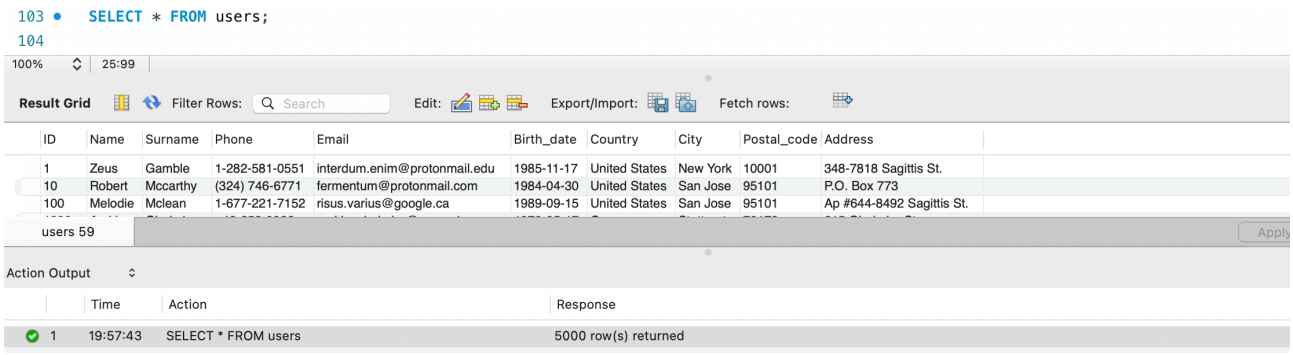
```
88  LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/downloads/european_users(1).csv'
89  INTO TABLE users
90  FIELDS TERMINATED BY ';'
91  ENCLOSED BY '"'
92  LINES TERMINATED BY '\n'
93  IGNORE 1 ROWS;
94
95  LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/downloads/american_users(1).csv'
96  INTO TABLE users
97  FIELDS TERMINATED BY ';'
98  ENCLOSED BY '"'
99  LINES TERMINATED BY '\n'
100 IGNORE 1 ROWS;
101
```

100% 28:83

Action Output

	Time	Action	Response
1	19:55:12	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0
2	19:55:22	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0

Vamos a ejecutar una consulta SELECT para asegurarnos de que los datos se cargan correctamente:



```
103  SELECT * FROM users;
104
```

100% 25:99

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

ID	Name	Surname	Phone	Email	Birth_date	Country	City	Postal_code	Address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	1985-11-17	United States	New York	10001	348-7818 Sagittis St.
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	1984-04-30	United States	San Jose	95101	P.O. Box 773
100	Melodie	Mclean	1-677-221-7152	risus.varius@google.ca	1989-09-15	United States	San Jose	95101	Ap #644-8492 Sagittis St.

users 59

Action Output

	Time	Action	Response
1	19:57:43	SELECT * FROM users	5000 row(s) returned

Como resultado de la consulta, podemos ver que cada valor coincide con la finalidad del campo y está representado por el valor correcto.

Transactions

El documento transactions.csv contiene todas las transacciones registradas por nuestro sistema. Contiene los siguientes campos con datos:

- ID,
- Card id
- Business id (Company_id),
- Timestamp,
- Amount,
- Declined,
- Product id,
- User id,
- Lat,
- Longitude;

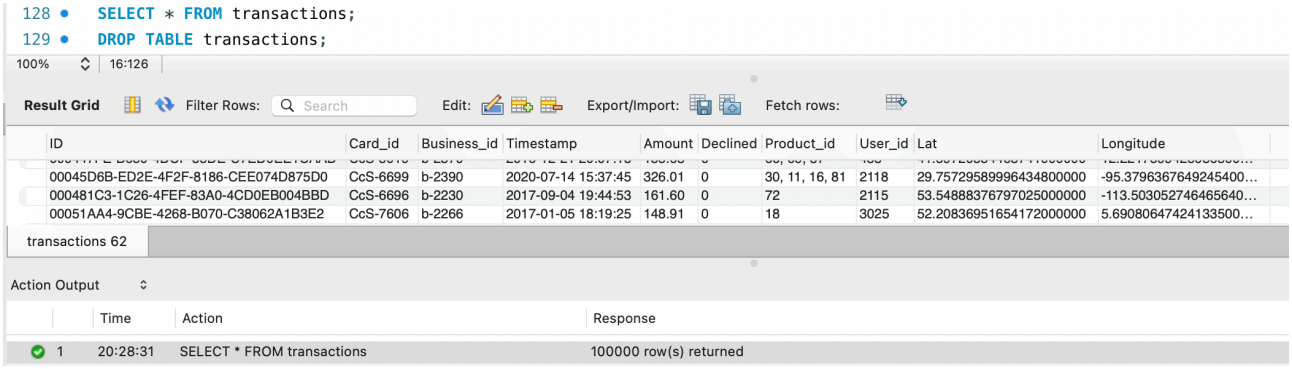
Como contiene información sobre todas las entidades de otras tablas de la base de datos, así como información sobre importes, fechas de transacciones - esta tabla será una tabla de hechos de nuestra base de datos. De acuerdo con otras tablas contiene FKs como Card_id, User_id, Business_id.

```
CREATE TABLE IF NOT EXISTS transactions(
        ID VARCHAR(36)PRIMARY KEY,
        Card_id VARCHAR (36),
        Business_id VARCHAR (36),
        Timestamp TIMESTAMP,
        Amount DECIMAL(10,2),
        Declined TINYINT(1),
        Product_id VARCHAR (36),
        User_id VARCHAR(36),
        Lat DECIMAL(25,20),
        Longitude DECIMAL(25,20),
FOREIGN KEY (User_id) REFERENCES users (ID),
FOREIGN KEY (Business_id) REFERENCES company (Company_ID),
FOREIGN KEY (Card_id) REFERENCES credit_cards (ID));
```

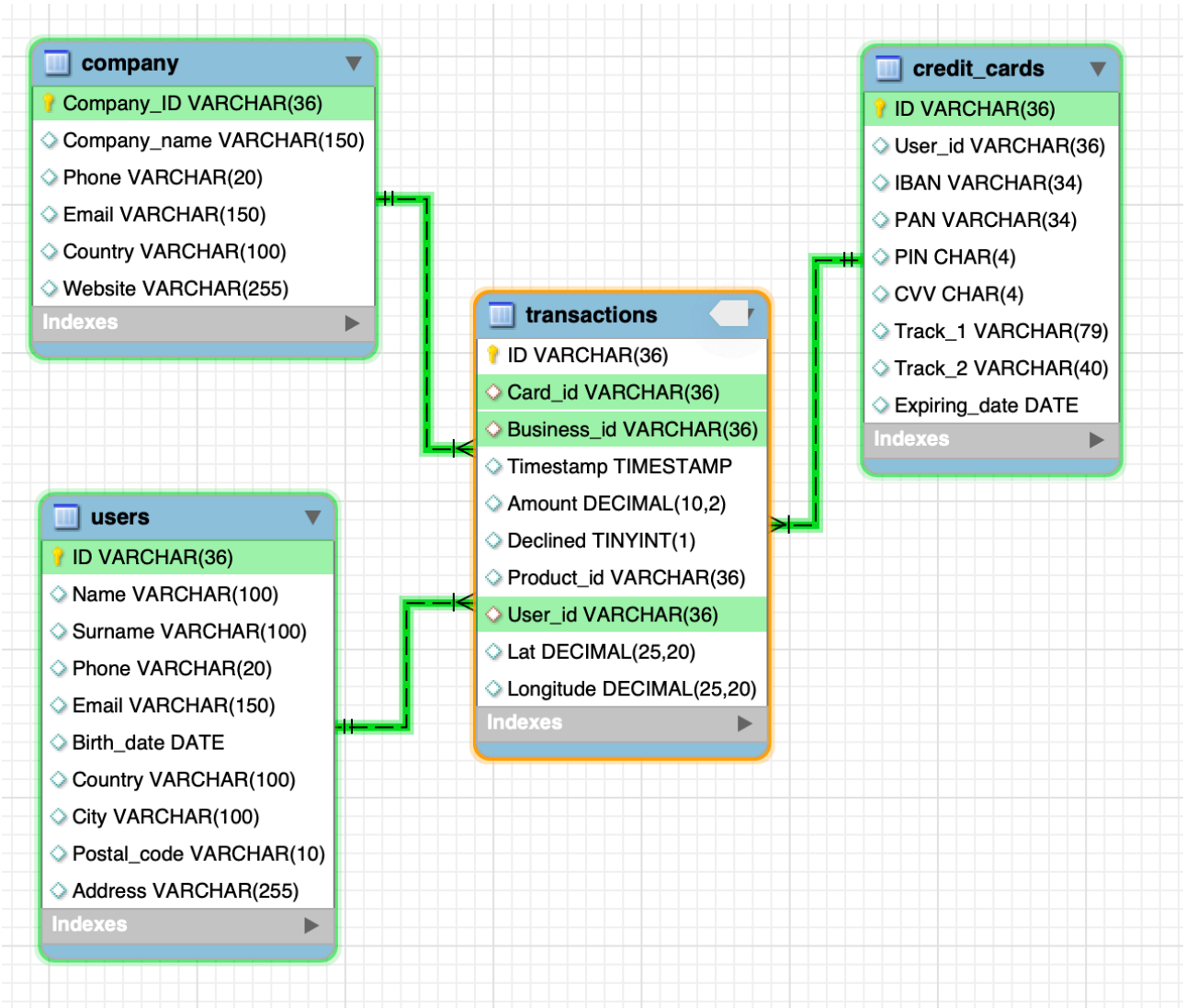
Resultado de la ejecución de la consulta:

He utilizado la siguiente consulta para importar datos a la tabla transacciones :

Vamos a ejecutar una consulta SELECT para asegurarnos de que los datos se cargan correctamente:



Como resultado de todas las consultas SQL ejecutadas obtenemos la siguiente base de datos:



Base de datos en esquema de estrella con cuatro tablas: credit_cards, user, company — las tablas de tipo dimensión, transaction — tabla de hechos de transacciones o entidad de hechos de transacciones registrados.

EJERCICIO 1
REALIZA UNA SUBCONSULTA QUE MUESTRE A TODOS LOS USUARIOS CON MÁS DE 80 TRANSACCIONES UTILIZANDO AL MENOS 2 TABLAS.

```
143 •  SELECT * FROM users
144    WHERE id IN
145        (SELECT user_id FROM transactions
146        GROUP BY user_id
147        HAVING COUNT(id) > 80);
148
```

100% 70:137

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

ID	Name	Surname	Phone	Email	Birth_date	Country	City	Postal_code	Address
185	Molly	Gilliam	0800 120 8023	donec@outlook.couk	1993-12-21	United Kingdom	London	EC1A 1BB	P.O. Box 202, 5638 Mi Rd.
289	Dxwgi	Hwcru	+98-309-8797	dxwgi.hwcru@example.com	1976-08-20	Germany	Stuttgart	70173	82 Hwcru Street
318	Bnyr	Astuw	+33-120-9644	bnyr.astuw@example.com	1974-05-03	Italy	Genoa	16100	53 Astuw Street
454	Sfzzoh	Xgvfridxs	+58-495-3945	sfzzoh.xgvfridxs@example.com	1962-08-28	Poland	Gdansk	80-001	52 Xgvfridxs Street
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

users 63

Action Output

	Time	Action	Response
1	20:38:09	SELECT * FROM users WHERE id IN (SELECT user_id FROM...	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.

```
140    -- Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.
141 •  SELECT round(AVG(transactions.amount),2) AS 'Coste medio',
142            credit_cards.IBAN
143    FROM credit_cards
144    JOIN transactions
145    ON transactions.card_id = credit_cards.id
146    JOIN company
147    ON transactions.business_id = company.company_id
148    WHERE company.company_name = 'Donec Ltd'
149    GROUP BY credit_cards.IBAN;
150
```

100% 21:137

Result Grid Filter Rows: Search Export:

Coste medio	IBAN
356.25	XX9114064011255863075868O5
142.96	SK9446370242474562577506
257.37	XX7767529178459529755556 40
139.59	XX4138273622897193049089 90
240.41	XX3477872460707696107803 08

Result 65

Action Output

	Time	Action	Response
1	20:42:28	SELECT round(AVG(transactions.amount),2) AS 'Coste medi...	371 row(s) returned

CREA UNA NUEVA TABLA QUE REFLEJE EL ESTADO DE LAS TARJETAS DE CRÉDITO BASADO EN SI LAS ÚLTIMAS TRES TRANSACCIONES FUERON DECLINADAS

```
153  CREATE TABLE IF NOT EXISTS credit_card_status (
154      id VARCHAR (36),
155      status VARCHAR (36));
156
```

100% 28:149

Action Output

	Time	Action	Response
1	10:58:45	CREATE TABLE IF NOT EXISTS credit_card_status (id VARC...	0 row(s) affected

PARA RELLENAR LA TABLA CON DATOS, UTILIZAMOS LA SIGUIENTE CONSULTA:

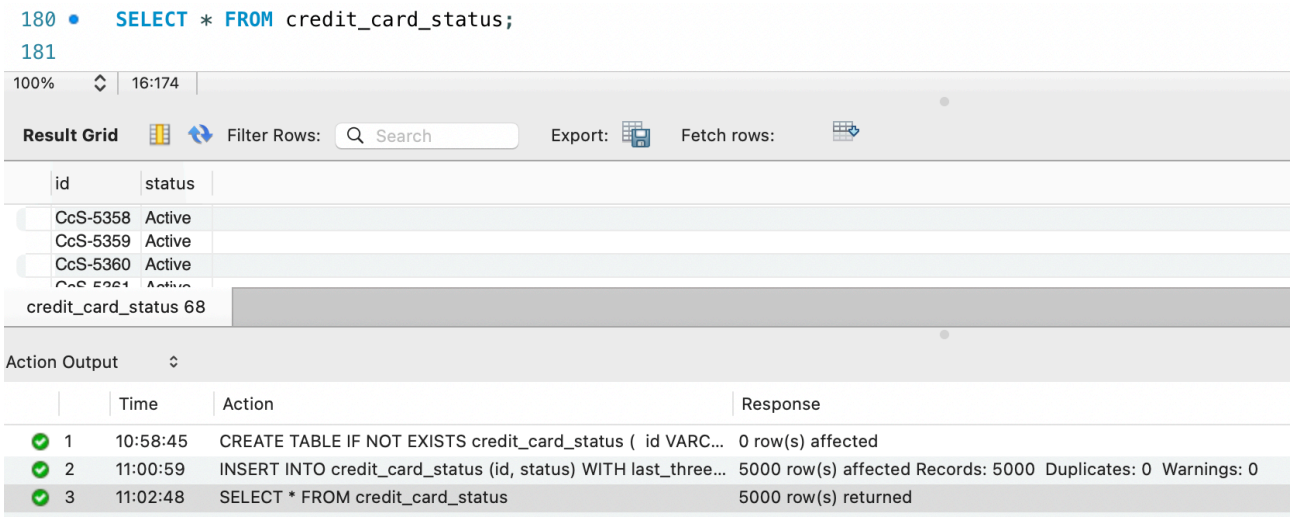
```
157  INSERT INTO credit_card_status (id, status)
158  WITH last_three AS (
159      SELECT
160          credit_cards.id,
161          transactions.timestamp,
162          transactions.declined,
163          ROW_NUMBER() OVER (
164              PARTITION BY credit_cards.id
165              ORDER BY transactions.timestamp DESC) AS rn
166      FROM credit_cards
167      JOIN transactions
168          ON credit_cards.id = transactions.card_id
169  )
170  SELECT
171      id,
172      CASE
173          WHEN SUM(declined) = 3 THEN 'Inactive'
174          ELSE 'Active'
175      END AS status
176  FROM last_three
177  WHERE rn <= 3
178  GROUP BY id;
179
```

100% 26:155

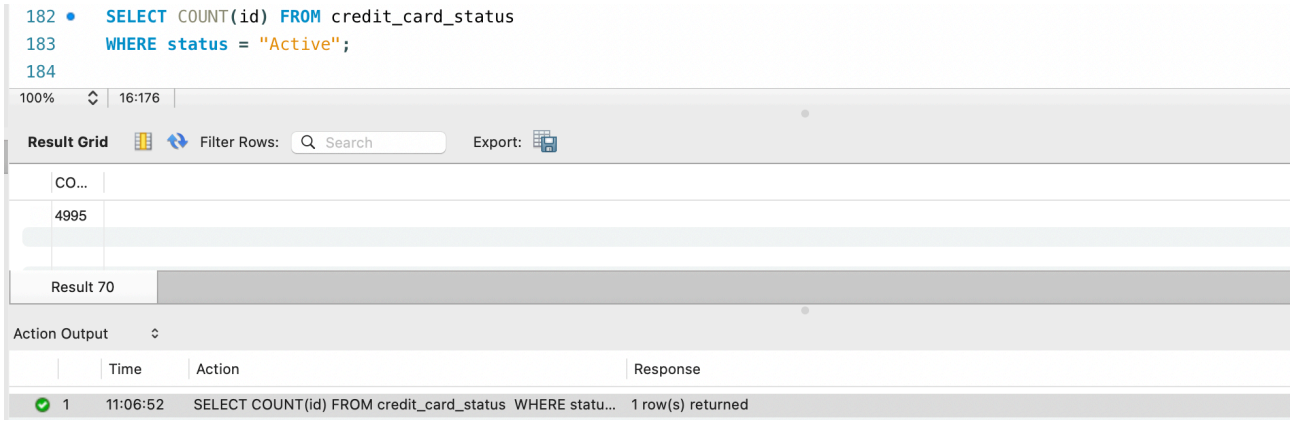
Action Output

	Time	Action	Response
1	10:58:45	CREATE TABLE IF NOT EXISTS credit_card_status (id VARC...	0 row(s) affected
2	11:00:59	INSERT INTO credit_card_status (id, status) WITH last_three...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

AHORA YA PODEMOS REALIZAR CUALQUIER CONSULTA RELACIONADA CON LA TABLA DE ESTADO DE LAS TARJETAS. POR EJEMPLO, CONSULTAR TODOS LOS DATOS DE LA TABLA:



EJERCICIO 1
¿CUÁNTAS TARJETAS ESTÁN ACTIVAS?



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.

Para crear nueva tabla e importar products.csv utilice la consulta:

```
37  CREATE TABLE IF NOT EXISTS products(
38      id VARCHAR(36)PRIMARY KEY,
39      product_name VARCHAR(150),
40      price DECIMAL (10,2),
41      colour VARCHAR(50),
42      weight DECIMAL (8,3),
43      warehouse_id Varchar(20));
44
```

100% 24:35

Action Output

	Time	Action	Response
1	11:24:57	CREATE TABLE IF NOT EXISTS products(id VARCHAR(36)P...	0 row(s) affected

Para importar correctamente los datos del archivo de origen, era necesario separar las columnas con un carácter que no se ajusta al tipo de datos especificado para la columna de precio. Para disponer de operaciones matemáticas al trabajar con la tabla en el futuro:

```
106  LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/downloads/products(2).csv'
107  INTO TABLE products
108  FIELDS TERMINATED BY ','
109  ENCLOSED BY '"'
110  LINES TERMINATED BY '\n'
111  IGNORE 1 ROWS;
112
```

100% 1:101

Action Output

	Time	Action	Response
1	11:24:57	CREATE TABLE IF NOT EXISTS products(id VARCHAR(36)P...	0 row(s) affected
2	11:25:45	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Asegúrese de comprobar que los datos se han cargado correctamente y corresponden a las filas y columnas especificados:

```
113  SELECT * FROM products;
```

100% 1:112

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

id	product_name	price	colour	weight	warehouse_id
100	south duel	40.43	#6d6d6d	3.000	WH--95
11	Karstark Dorne	49.70	#141414	2.700	WH--6
12	duel Direwolf	181.60	#a8a8a8	2.100	WH--7

products 73

Action Output

	Time	Action	Response
1	11:24:57	CREATE TABLE IF NOT EXISTS products(id VARCHAR(36)P...	0 row(s) affected
2	11:25:45	LOAD DATA LOCAL INFILE '/Users/ekaterinasorokopudova/do...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
3	11:28:58	SELECT * FROM products	100 row(s) returned

Como parte de esta tarea, necesitamos normalizar la base de datos y además crear una relación muchos a muchos entre dos tablas de transacciones y productos.

La normalización es necesaria porque en la versión actual de la tabla de transacciones el campo «ID de producto» contiene más que un ID por una transacción.

Para crear una relación muchos a muchos en SQL debemos utilizar una tercera tabla que contendrá datos atómicos. Para crearla, necesitamos implementar una consulta:

```sql
CREATE TABLE IF NOT EXISTS transaction_product (
transaction_id VARCHAR(36),
product_id VARCHAR(36),
FOREIGN KEY (product_id) REFERENCES products(ID),
FOREIGN KEY (transaction_id) REFERENCES transactions(ID));

```

100% 17:197

Action Output

	Time	Action	Response
1	11:32:24	CREATE TABLE IF NOT EXISTS transaction_product (transa...	0 row(s) affected

Para rellenar la tabla con datos, utilizamos la siguiente consulta:

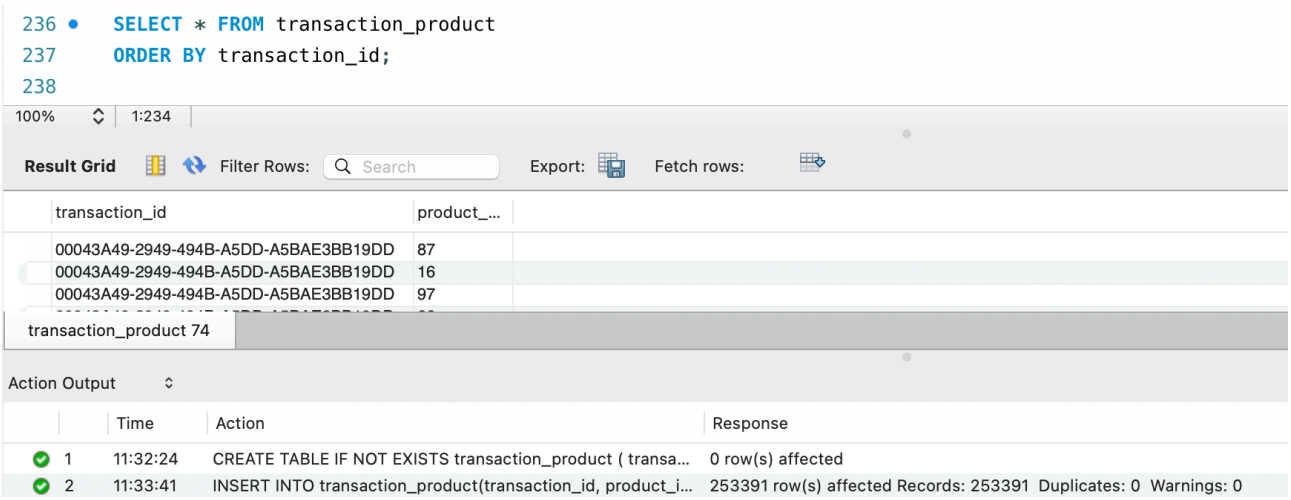
```sql
INSERT INTO transaction_product(transaction_id, product_id)
WITH RECURSIVE split_ids AS (
  SELECT
    id AS transaction_id,
    TRIM(SUBSTRING_INDEX(product_id, ',', 1)) AS product_id,
    SUBSTRING(product_id, LENGTH(SUBSTRING_INDEX(product_id, ',', 1)) + 2) AS rest
  FROM transactions

  UNION ALL

  SELECT
    transaction_id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)),
    SUBSTRING(rest, LENGTH(SUBSTRING_INDEX(rest, ',', 1)) + 2)
  FROM split_ids
  WHERE rest != ''
)
SELECT transaction_id, product_id FROM split_ids;

```

100% 83:221

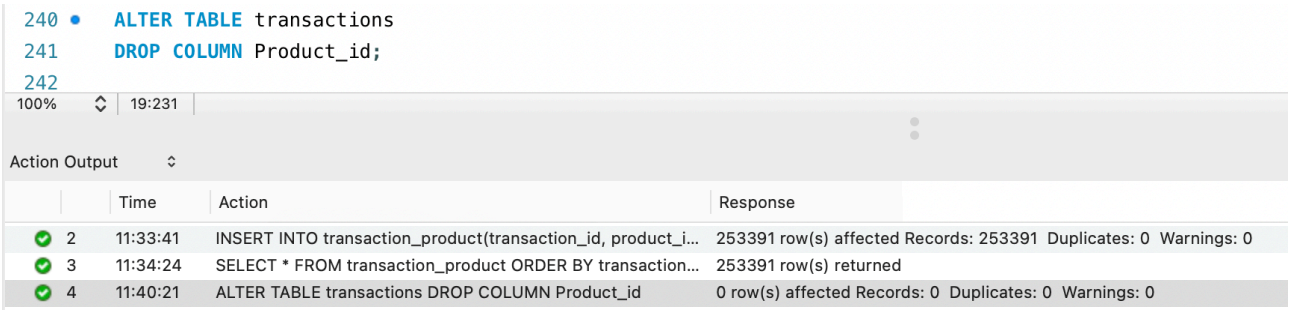
Action Output

	Time	Action	Response
1	11:32:24	CREATE TABLE IF NOT EXISTS transaction_product (transa...	0 row(s) affected
2	11:33:41	INSERT INTO transaction_product(transaction_id, product_i...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

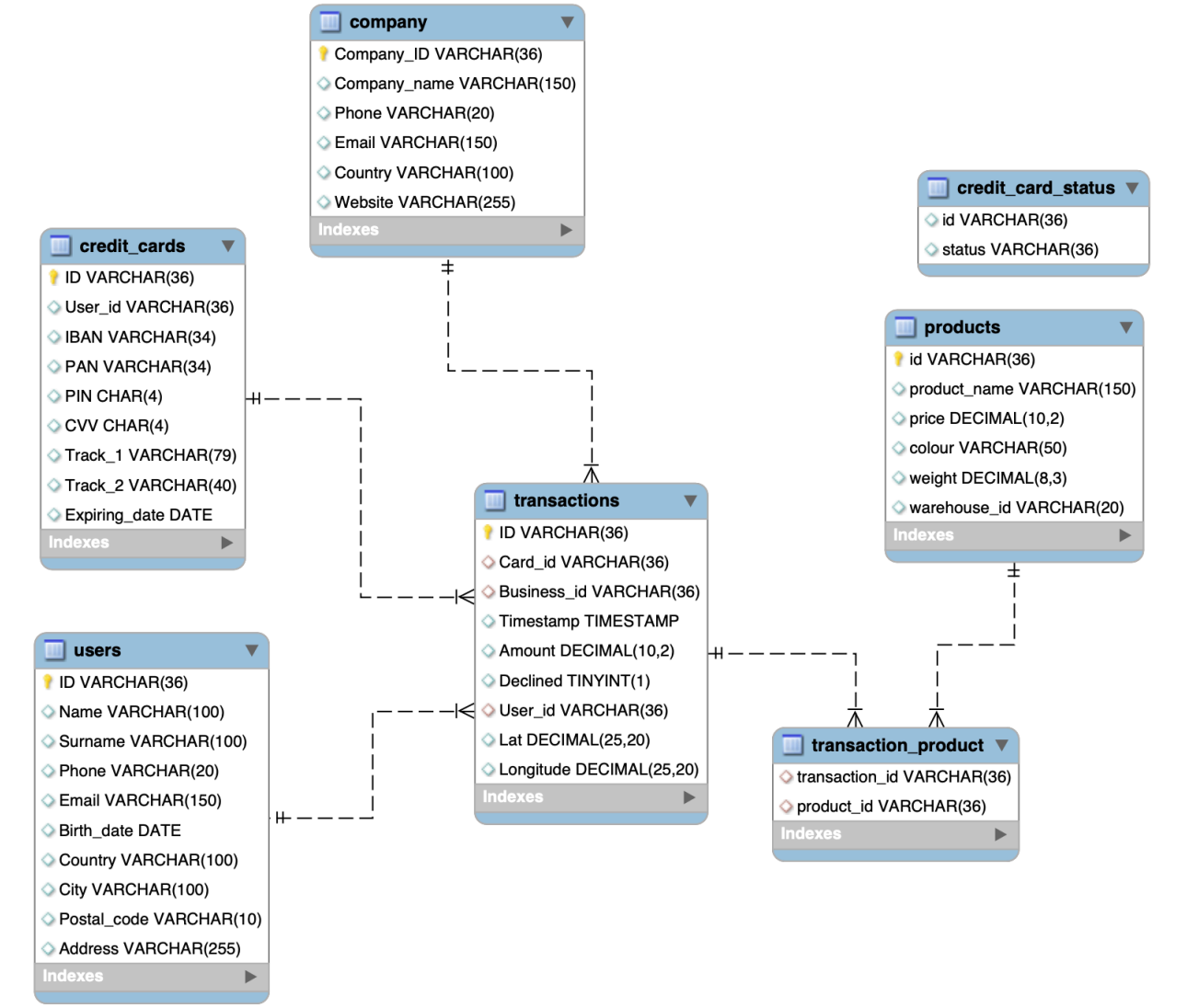
El resultado de la ejecución de la consulta es la creación de una tabla con un id de producto para cada transacción, y si hay más de un id para una transacción, el valor se traslada a la fila siguiente:



A continuación, para mantener la coherencia de los datos y normalizar la base de datos, debemos eliminar la columna Product_ID de la tabla de transacciones:



Al final, la base de datos tiene el siguiente aspecto:



EJERCICIO 1
NECESITAMOS CONOCER EL NÚMERO DE VECES QUE SE HA VENDIDO CADA PRODUCTO.

```
244 •   SELECT COUNT(product_name) AS 'cantidad',
245             product_name
246             FROM products
247     JOIN transaction_product
248         ON products.ID = transaction_product.product_id
249     GROUP BY id;
```

100% | 24:241

Result Grid | Filter Rows: Search | Export:

cantidad	product_name
2517	south duel
2573	Karstark Dorne
2558	duel Direwolf

Result 76

Action Output

	Time	Action	Response
1	11:46:53	SELECT COUNT(product_name) AS 'cantidad', product_na...	100 row(s) returned