

SPRINT _ 4

CREACION DE BASE DE DATOS

A partir de algunos archivos CSV diseñarás y crearás tu base de datos.

Nivel 1

Descargue los archivos CSV, estúdielos y diseñe una base de datos con un esquema estrella que contenga al menos 4 tablas desde la que se puedan hacer las siguientes consultas:

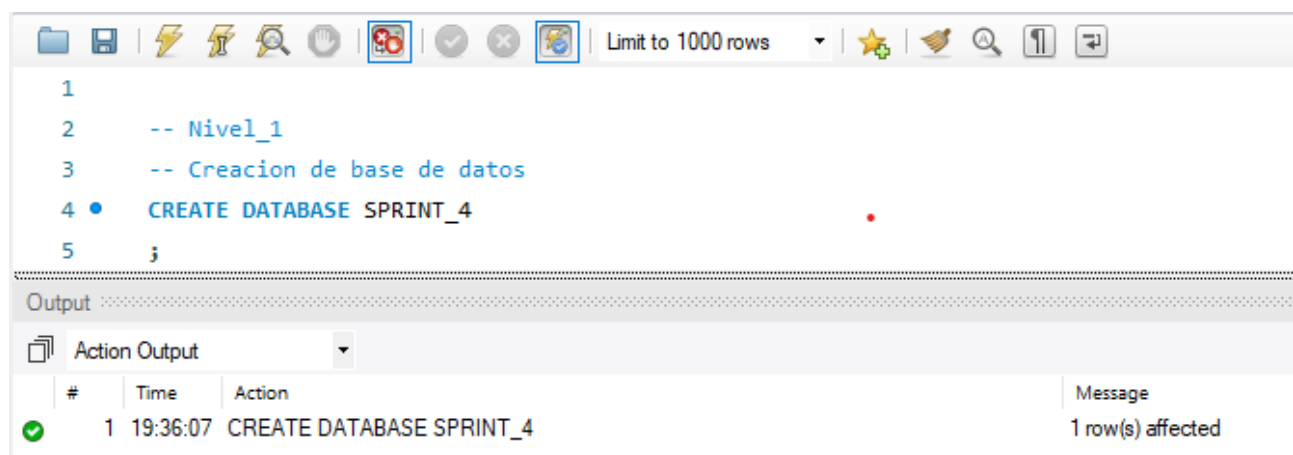
Creación de la base de datos.

Se crea la base de datos SPRINT _ 4 .

La base de datos estará diseñada con un esquema en estrella.

La base de datos constará de una tabla de hechos (transactions) y tres tablas de dimensiones (users, companies, credit_cards).

Creación de la base de datos.



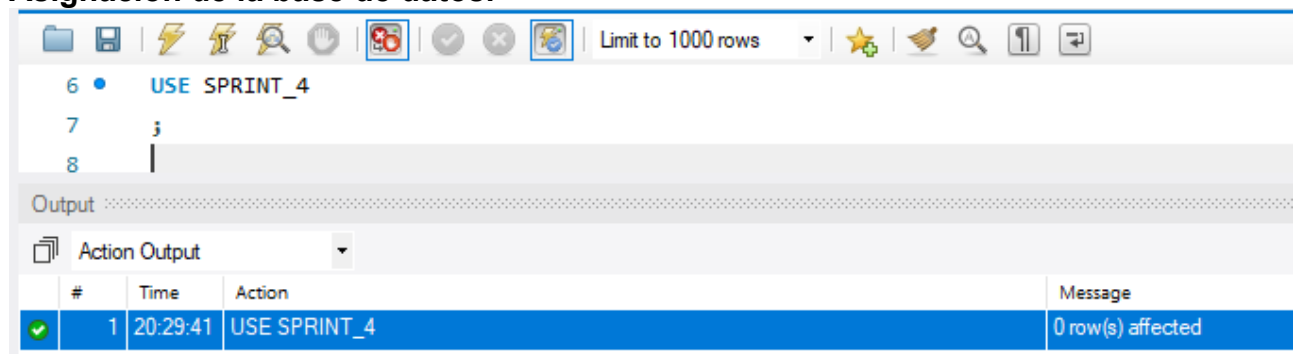
The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. The main editor area contains the following SQL code:

```
1
2  -- Nivel_1
3  -- Creacion de base de datos
4  • CREATE DATABASE SPRINT_4
5  ;
```

Below the editor is an "Output" pane with a dropdown menu set to "Action Output". It displays a table with the following data:

#	Time	Action	Message
✓ 1	19:36:07	CREATE DATABASE SPRINT_4	1 row(s) affected

Asignación de la base de datos.



The screenshot shows the same SQL IDE interface. The main editor area contains the following SQL code:

```
6  • USE SPRINT_4
7  ;
8  |
```

Below the editor is an "Output" pane with a dropdown menu set to "Action Output". It displays a table with the following data:

#	Time	Action	Message
✓ 1	20:29:41	USE SPRINT_4	0 row(s) affected

Creación de la tabla users.

El datatype de los campos se ha asignado en función del tipo de datos existentes en los archivos csv.

Los campos tipo texto se han asignado como VARCHAR(*).

El campo "id" se ha asignado como INT.

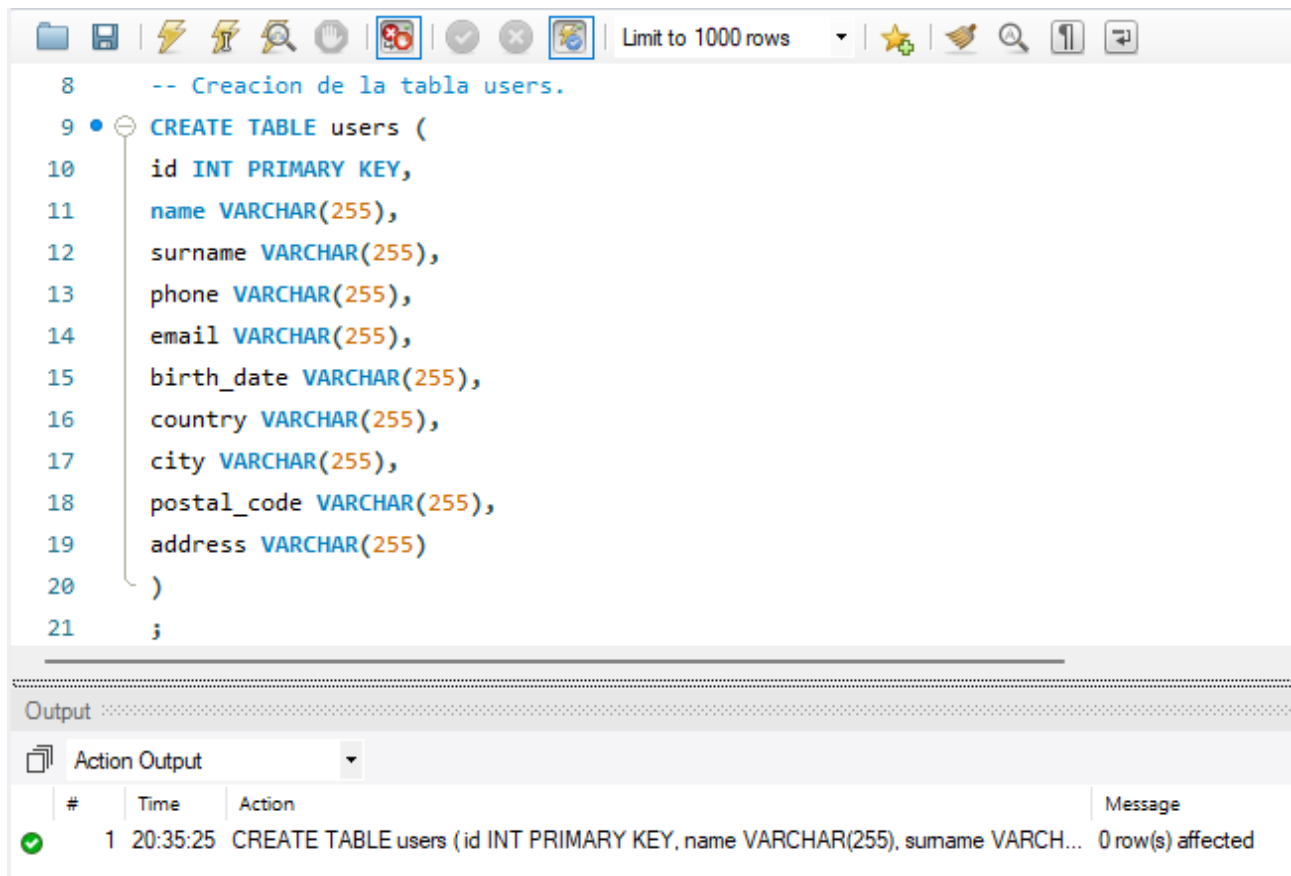
La columna "birth_date" se define como VARCHAR para evitar posibles conflictos de formato en la introducción de datos si fuese definida como tipo DATE.

El tipo de datos puede ser modificado posteriormente para adaptarlo al formato DATE.

Se ha asignado el campo "id" como PRIMARY KEY.

(*) Los campos tipo texto se han asignado de forma generica como VARCHAR(255). Esto permite que los estos campos acepten una gran variedad de longitud de datos sin que afecte al rendimiento del sistema.

Los campos VARCHAR ocupan una cantidad de memoria variable en función de la longitud de los datos y un byte por la definición de la longitud máxima del campo si esta es menor o igual a 255. Si la definición de la longitud máxima del campo fuese superior a 255 ocuparia 2 bytes de memoria.




```
8  -- Creacion de la tabla users.
9  CREATE TABLE users (
10     id INT PRIMARY KEY,
11     name VARCHAR(255),
12     surname VARCHAR(255),
13     phone VARCHAR(255),
14     email VARCHAR(255),
15     birth_date VARCHAR(255),
16     country VARCHAR(255),
17     city VARCHAR(255),
18     postal_code VARCHAR(255),
19     address VARCHAR(255)
20 )
21 ;
```

Output

Action Output

#	Time	Action	Message
✓ 1	20:35:25	CREATE TABLE users (id INT PRIMARY KEY, name VARCHAR(255), surname VARCH...	0 row(s) affected

Comprobación de la tabla users.

Limit to 1000 rows

```
22  -- Comprobacion de la tabla users.
23  •  show columns
24    from users
25
26
```

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

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	
	name	varchar(255)	YES		NULL	
	surname	varchar(255)	YES		NULL	
	phone	varchar(255)	YES		NULL	
	email	varchar(255)	YES		NULL	
	birth_date	varchar(255)	YES		NULL	
	country	varchar(255)	YES		NULL	
	city	varchar(255)	YES		NULL	
	postal_code	varchar(255)	YES		NULL	
	address	varchar(255)	YES		NULL	

Result 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	21:06:02	show columns from users	10 row(s) returned

Creación de la tabla companies.

El datatype de los campos se ha asignado en función del tipo de datos existentes en los archivos csv.

Todos los campos de la tabla se han definido como VARCHAR.

Se ha asignado el campo "company_id" como PRIMARY_KEY.

The screenshot shows a SQL editor with the following code:

```
28 -- Creacion de la tabla companies.
29 CREATE TABLE companies (
30     company_id VARCHAR(255) PRIMARY KEY
31     , company_name VARCHAR(255)
32     , phone VARCHAR(255)
33     , email VARCHAR(255)
34     , country VARCHAR(255)
35     , website VARCHAR(255)
36 )
37 ;
```

Below the editor is an "Output" section with a table showing the execution of the SQL command:

#	Time	Action	Message
1	21:15:52	CREATE TABLE companies (company_id VARCHAR(255) PRIMARY KEY , company_n...	0 row(s) affected

Comprobación de la tabla companies.

The screenshot shows a SQL editor with the following code:

```
39 -- Comprobacion de la tabla companies.
40 SHOW COLUMNS
41 FROM companies
42 ;
43
```

Below the editor is a "Result Grid" showing the table structure:

Field	Type	Null	Key	Default	Extra
company_id	varchar(255)	NO	PRI	NULL	
company_name	varchar(255)	YES		NULL	
phone	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
country	varchar(255)	YES		NULL	
website	varchar(255)	YES		NULL	

Below the result grid is an "Output" section with a table showing the execution of the SQL command:

#	Time	Action	Message
1	21:20:38	SHOW COLUMNS FROM companies	6 row(s) returned

Creación de la tabla `credit_cards`.

El datatype de los campos se ha asignado en función del tipo de datos existentes en el archivo csv.

Los campos tipo texto se han asignado como `VARCHAR`.

La columna `"user_id"` se define como `INT`.

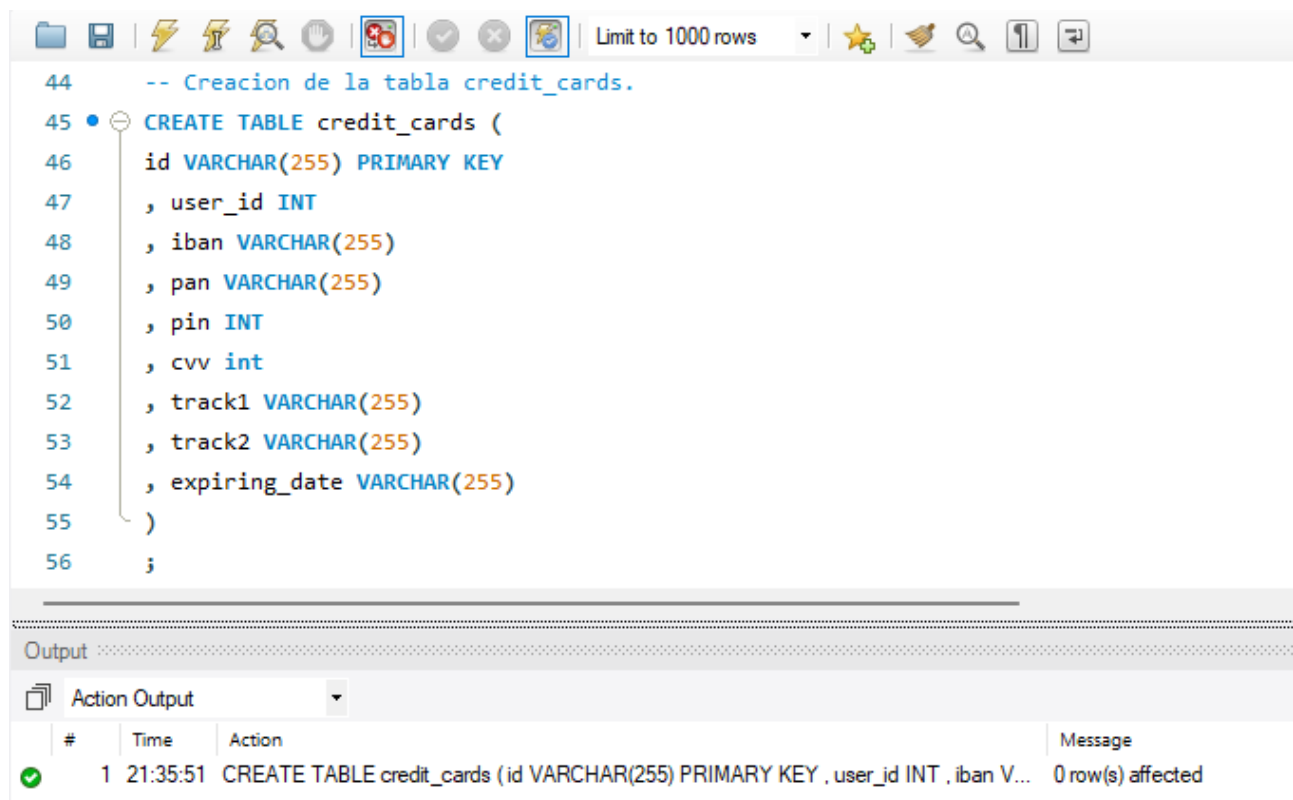
La columna `"pin"` se define como `INT`.

La columna `"cvv"` se define como `INT`.

La columna `"expiring_date"` se define como `VARCHAR` para evitar posibles conflictos de formato en la introducción de datos si fuese definida como tipo `DATE`.

El tipo de datos puede ser modificado posteriormente para adaptarlo al formato `DATE`.

Se ha asignado el campo `"id"` como `PRIMARY KEY`.




```
44  -- Creacion de la tabla credit_cards.
45  CREATE TABLE credit_cards (
46      id VARCHAR(255) PRIMARY KEY
47      , user_id INT
48      , iban VARCHAR(255)
49      , pan VARCHAR(255)
50      , pin INT
51      , cvv int
52      , track1 VARCHAR(255)
53      , track2 VARCHAR(255)
54      , expiring_date VARCHAR(255)
55  )
56  ;
```

Output




Action Output

#	Time	Action	Message
✓ 1	21:35:51	CREATE TABLE credit_cards (id VARCHAR(255) PRIMARY KEY , user_id INT , iban V...	0 row(s) affected

Comprobación de la tabla credit_cards.

 Limit to 1000 rows


```
58 -- Comprobacion de la tabla credit_cards.
59 • SHOW COLUMNS
60 FROM credit_cards
61 ;
62
```

Result Grid  Filter Rows: Export:  Wrap Cell Content: 

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(255)	NO	PRI	NULL	
	user_id	int	YES		NULL	
	iban	varchar(255)	YES		NULL	
	pan	varchar(255)	YES		NULL	
	pin	int	YES		NULL	
	cvv	int	YES		NULL	
	track1	varchar(255)	YES		NULL	
	track2	varchar(255)	YES		NULL	
	expiring_date	varchar(255)	YES		NULL	

Result 6 ×

Output

 Action Output

#	Time	Action	Message
✓ 1	21:40:13	SHOW COLUMNS FROM credit_cards	9 row(s) returned

Creación de la tabla transactions.

Se crea la tabla "transactions".

El datatype de los campos se ha asignado en función del tipo de datos existentes en el archivo csv.

Los campos tipo texto se han asignado como VARCHAR.

El campo "timestamp" se ha definido como TIMESTAMP.

El campo "amount" se ha definido como "DECIMAL".

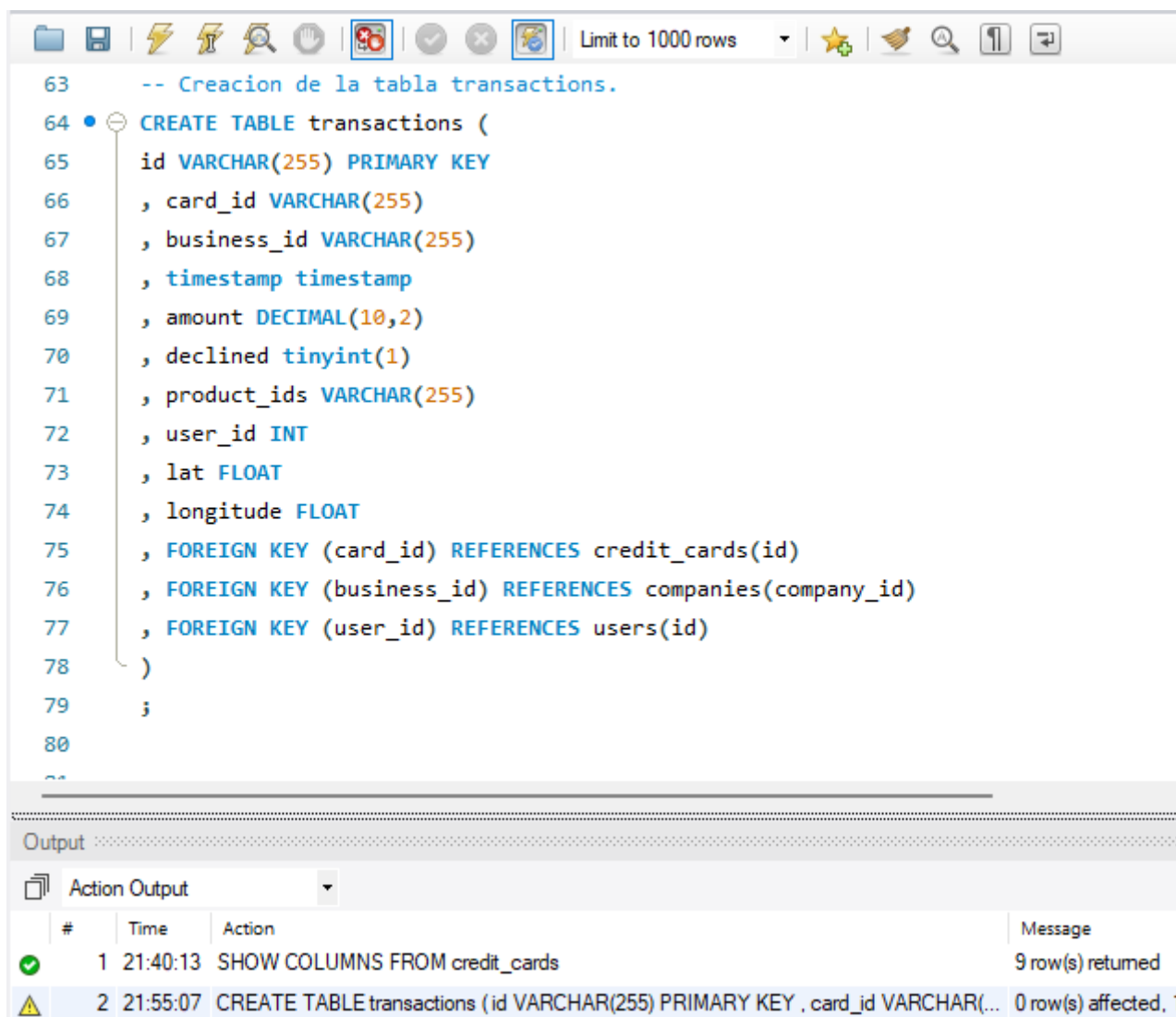
El campo "declined" se ha definido como TINYINT.

El campo "user_id" se ha definido como INT.

Los campos "lat" y "longitude" se han definido como FLOAT.

Se crean las foreign key de la tabla de hechos "transactions" referenciadas a las primary key de las tablas de dimensiones "users", "companies" y "credit_cards".

Con esto se establecen unas relaciones 1:N entre las tablas de dimensiones ("users", "companies" y "credit_cards") y la tabla de hechos "transactions".





The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main area displays SQL code for creating the 'transactions' table. The code includes field definitions with data types and constraints, and foreign key references to 'credit_cards', 'companies', and 'users'. The bottom section, titled 'Output', shows the execution results. The first action, 'SHOW COLUMNS FROM credit_cards', returned 9 rows. The second action, 'CREATE TABLE transactions', affected 0 rows.




```
63  -- Creacion de la tabla transactions.
64  CREATE TABLE transactions (
65      id VARCHAR(255) PRIMARY KEY
66      , card_id VARCHAR(255)
67      , business_id VARCHAR(255)
68      , timestamp timestamp
69      , amount DECIMAL(10,2)
70      , declined tinyint(1)
71      , product_ids VARCHAR(255)
72      , user_id INT
73      , lat FLOAT
74      , longitude FLOAT
75      , FOREIGN KEY (card_id) REFERENCES credit_cards(id)
76      , FOREIGN KEY (business_id) REFERENCES companies(company_id)
77      , FOREIGN KEY (user_id) REFERENCES users(id)
78  )
79  ;
80
```

#	Time	Action	Message
1	21:40:13	SHOW COLUMNS FROM credit_cards	9 row(s) returned
2	21:55:07	CREATE TABLE transactions (id VARCHAR(255) PRIMARY KEY , card_id VARCHAR(255) , business_id VARCHAR(255) , timestamp timestamp , amount DECIMAL(10,2) , declined tinyint(1) , product_ids VARCHAR(255) , user_id INT , lat FLOAT , longitude FLOAT , FOREIGN KEY (card_id) REFERENCES credit_cards(id) , FOREIGN KEY (business_id) REFERENCES companies(company_id) , FOREIGN KEY (user_id) REFERENCES users(id)) ;	0 row(s) affected, 1

Comprobación de la tabla transactions.

Limit to 1000 rows


```
81  -- Comprobacion de la tabla transactions.
82  •  SHOW COLUMNS
83  FROM transactions
84  ;
85
```

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

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(255)	NO	PRI	NULL	
	card_id	varchar(255)	YES	MUL	NULL	
	business_id	varchar(255)	YES	MUL	NULL	
	timestamp	timestamp	YES		NULL	
	amount	decimal(10,2)	YES		NULL	
	declined	tinyint(1)	YES		NULL	
	product_ids	varchar(255)	YES		NULL	
	user_id	int	YES	MUL	NULL	
	lat	float	YES		NULL	
	longitude	float	YES		NULL	

Result 8 ×

Output

 Action Output

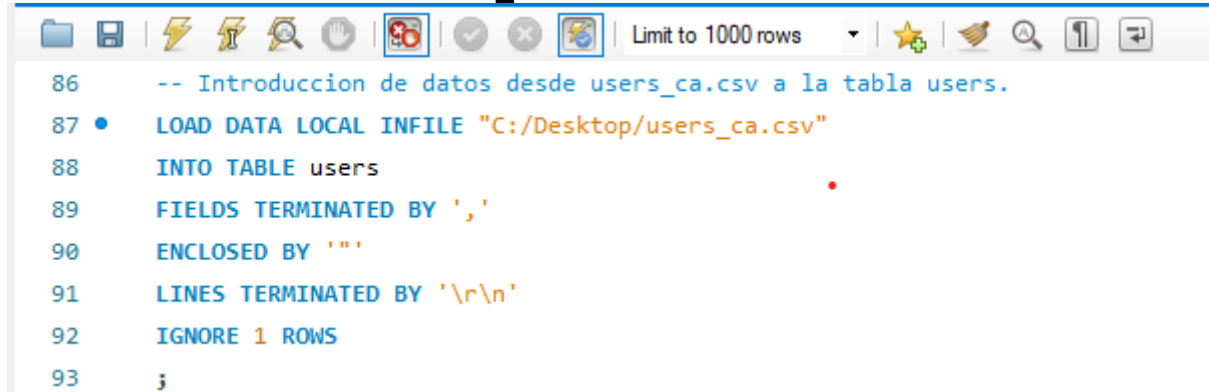
#	Time	Action	Message
✓ 1	21:58:51	SHOW COLUMNS FROM transactions	10 row(s) returned

Introducción de datos en las tablas.

Se introducen en las tablas los datos correspondientes desde los archivos csv.

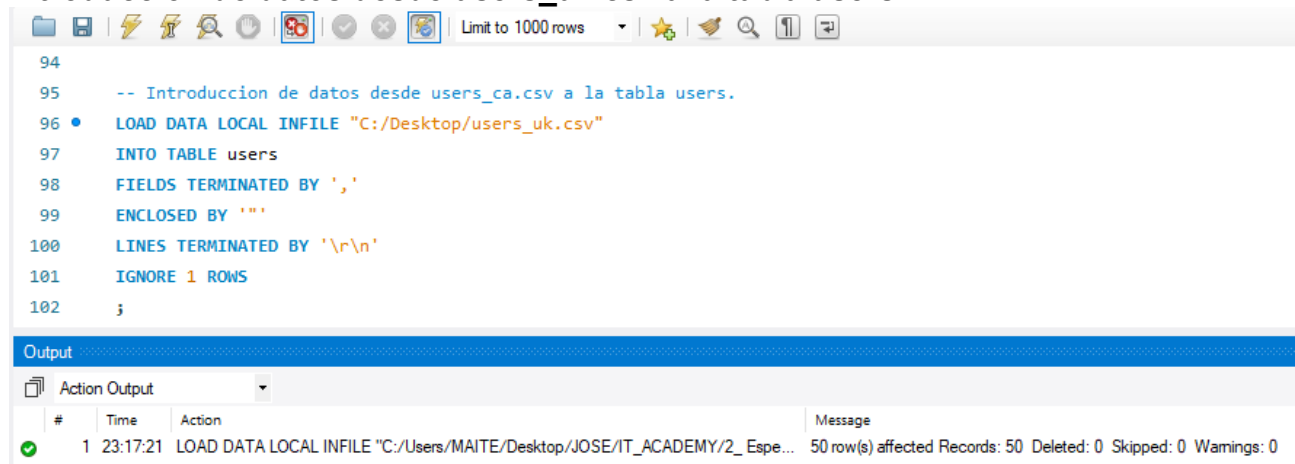
En la tabla users se introducen los datos de los archivos users_ca, users_uk y users_usa. Con esto se unifican los datos de los usuarios en una única tabla.

Introducción de datos desde users_ca.csv a la tabla users.



```
86  -- Introduccion de datos desde users_ca.csv a la tabla users.
87  •  LOAD DATA LOCAL INFILE "C:/Desktop/users_ca.csv"
88      INTO TABLE users
89      FIELDS TERMINATED BY ','
90      ENCLOSED BY '"'
91      LINES TERMINATED BY '\r\n'
92      IGNORE 1 ROWS
93  ;
```

Introducción de datos desde users_uk.csv a la tabla users.

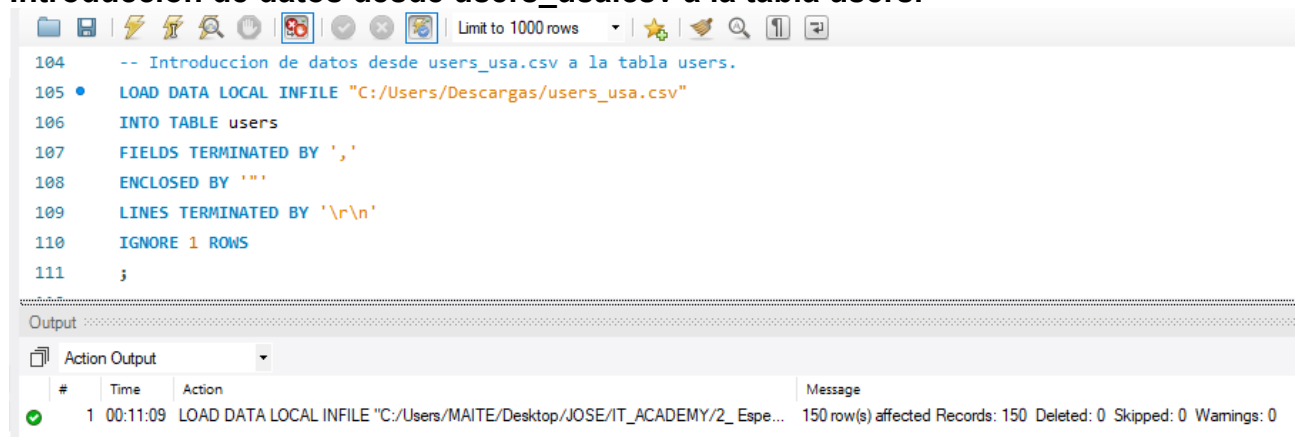


```
94
95  -- Introduccion de datos desde users_ca.csv a la tabla users.
96  •  LOAD DATA LOCAL INFILE "C:/Desktop/users_uk.csv"
97      INTO TABLE users
98      FIELDS TERMINATED BY ','
99      ENCLOSED BY '"'
100     LINES TERMINATED BY '\r\n'
101     IGNORE 1 ROWS
102  ;
```

Output

#	Time	Action	Message
1	23:17:21	LOAD DATA LOCAL INFILE "C:/Users/MAITE/Desktop/JOSE/IT_ACADEMY/2_Espe...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0

Introduccion de datos desde users_usa.csv a la tabla users.



```
104  -- Introduccion de datos desde users_usa.csv a la tabla users.
105  •  LOAD DATA LOCAL INFILE "C:/Users/Descargas/users_usa.csv"
106      INTO TABLE users
107      FIELDS TERMINATED BY ','
108      ENCLOSED BY '"'
109      LINES TERMINATED BY '\r\n'
110      IGNORE 1 ROWS
111  ;
```

Output

#	Time	Action	Message
1	00:11:09	LOAD DATA LOCAL INFILE "C:/Users/MAITE/Desktop/JOSE/IT_ACADEMY/2_Espe...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0

Introducción de datos desde companies.csv a la tabla companies.



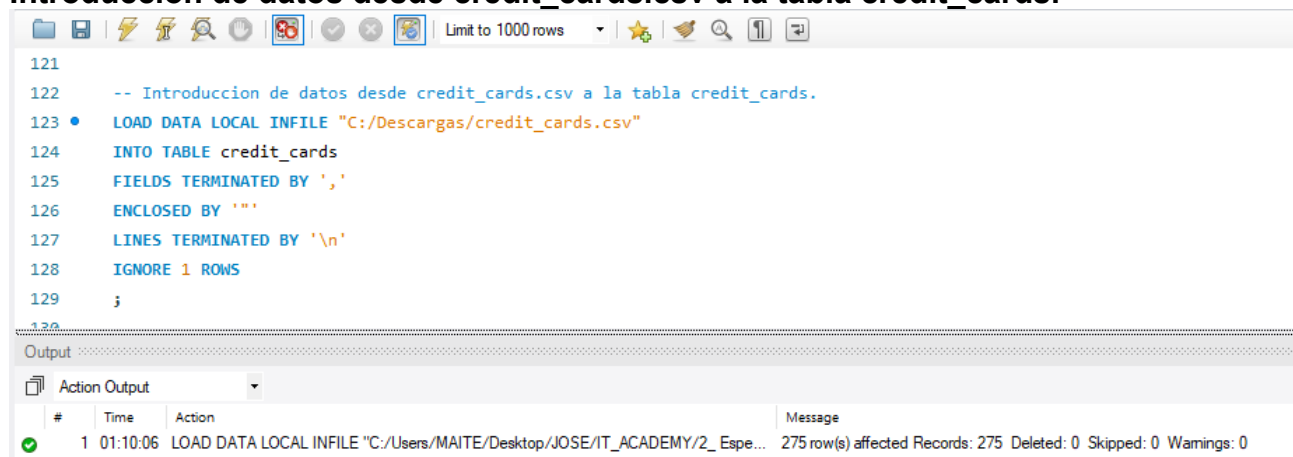
The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
113 -- Introduccion de datos desde companies.csv a la tabla companies.
114 • LOAD DATA LOCAL INFILE "C:/Descargas/companies.csv"
115 INTO TABLE companies
116 FIELDS TERMINATED BY ','
117 ENCLOSED BY '"'
118 LINES TERMINATED BY '\r\n'
119 IGNORE 1 ROWS
120 ;
```

The Output tab is selected, showing the execution results:

#	Time	Action	Message
1	00:34:29	LOAD DATA LOCAL INFILE "C:/Users/MAITE/Desktop/JOSE/IT_ACADEMY/2_Espe...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Introducción de datos desde credit_cards.csv a la tabla credit_cards.



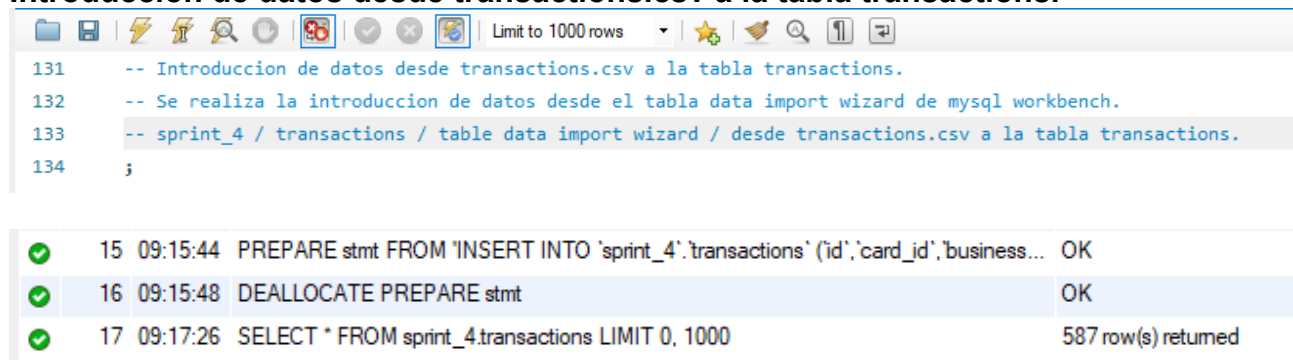
The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
121
122 -- Introduccion de datos desde credit_cards.csv a la tabla credit_cards.
123 • LOAD DATA LOCAL INFILE "C:/Descargas/credit_cards.csv"
124 INTO TABLE credit_cards
125 FIELDS TERMINATED BY ','
126 ENCLOSED BY '"'
127 LINES TERMINATED BY '\n'
128 IGNORE 1 ROWS
129 ;
```

The Output tab is selected, showing the execution results:

#	Time	Action	Message
1	01:10:06	LOAD DATA LOCAL INFILE "C:/Users/MAITE/Desktop/JOSE/IT_ACADEMY/2_Espe..."	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0

Introduccion de datos desde transactions.csv a la tabla transactions.



The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
131 -- Introduccion de datos desde transactions.csv a la tabla transactions.
132 -- Se realiza la introduccion de datos desde el tabla data import wizard de mysql workbench.
133 -- sprint_4 / transactions / table data import wizard / desde transactions.csv a la tabla transactions.
134 ;
```

The Output tab is selected, showing the execution results:

✓	15	09:15:44	PREPARE stmt FROM 'INSERT INTO `sprint_4`.`transactions` (`id`,`card_id`,`business...	OK
✓	16	09:15:48	DEALLOCATE PREPARE stmt	OK
✓	17	09:17:26	SELECT * FROM sprint_4.transactions LIMIT 0, 1000	587 row(s) returned

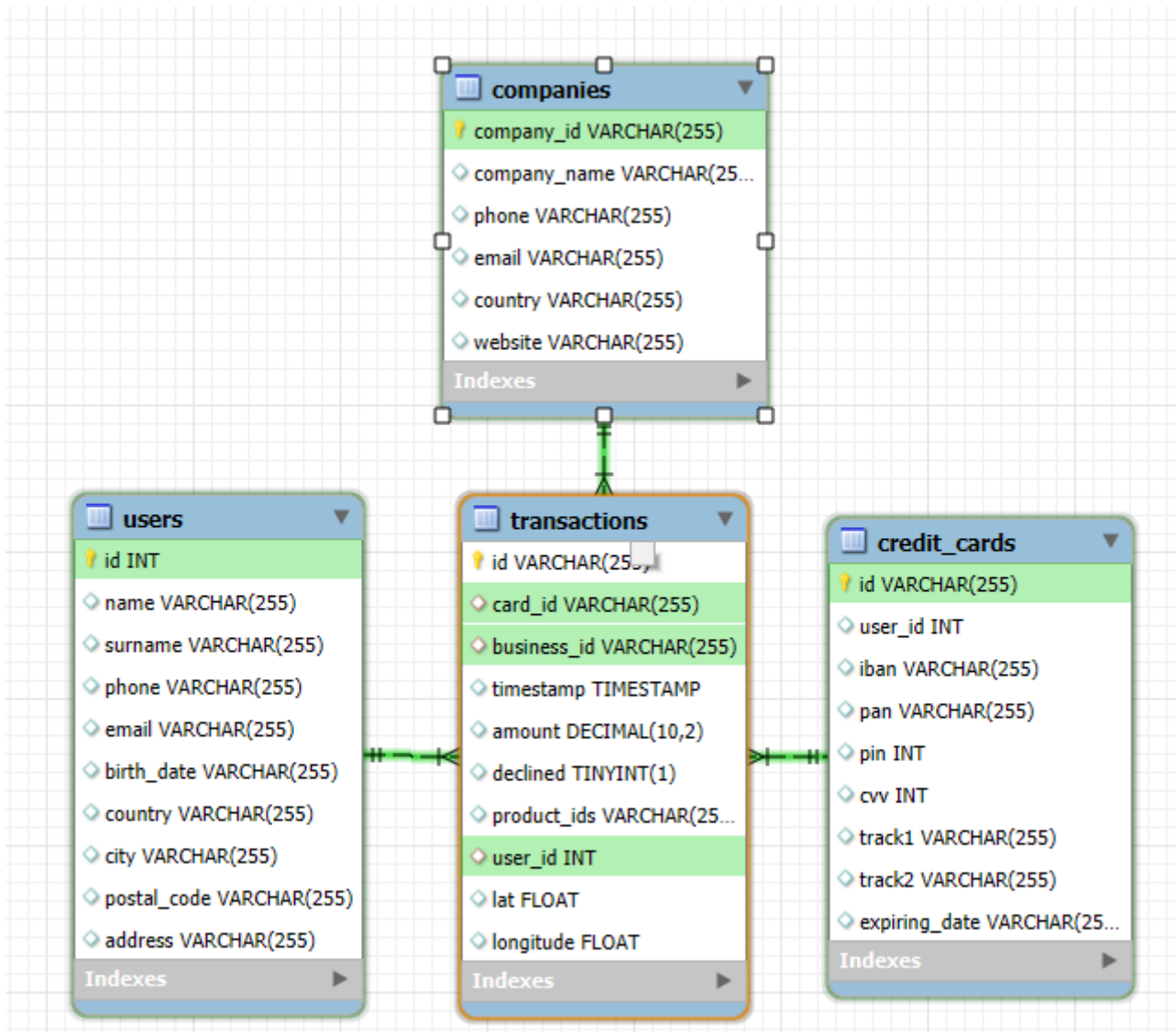
Esquema de la base de datos.

Se ha creado una base de datos con un modelo en estrella.

La tabla de hechos es la tabla "transactions".

Las tablas de dimensiones son las tablas: "users", "companies" y "credit_cards".

Las relaciones establecidas son de 1:N desde las tablas de dimensiones a la tabla de hechos.



Nivel 1.

Ejercicio 1

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

```
137  -- Ejercicio 1
138  /*
139   Haz un subconsulta que muestre a todos los usuarios con más de 30 transacciones usando al menos 2 tablas.
140  */
141  • SELECT transactions.user_id
142      , users.name
143      , users.surname
144      , COUNT(transactions.id) AS num_transactions
145  FROM transactions
146  JOIN users ON transactions.user_id = users.id
147  GROUP BY user_id
148  HAVING num_transactions > 30
149  ORDER BY num_transactions DESC
150  ;
```

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

	user_id	name	surname	num_transactions
▶	272	Hedwig	Gilbert	76
	267	Ocean	Nelson	52
	275	Kenyon	Hartman	48
	92	Lynn	Riddle	39

Result 4 ×

Output

Action Output

#	Time	Action	Message
✓ 1	09:54:57	SELECT transactions.user_id , users.name , users.surname , COUNT(transactions.id) A...	4 row(s) returned

Nivel 1

Ejercicio 2

Mostrar el promedio de la tarjeta de crédito por IBAN en la empresa Donec Ltd, utilizar al menos 2 tablas.

```
152 -- Nivel 1
153 -- Ejercicio 2
154 /*
155  Mostrar el promedio de amount de la tarjeta de crédito por IBAN en la empresa Donec Ltd, utilizar al menos 2 tablas.
156 */
157 • SELECT companies.company_name AS company
158     , credit_cards.iban AS iban
159     , AVG(transactions.amount) AS average_transaction
160 FROM transactions
161 JOIN credit_cards ON transactions.card_id = credit_cards.id
162 JOIN companies ON transactions.business_id = companies.company_id
163 WHERE companies.company_name = 'Donec Ltd'
164 GROUP BY iban, company
165 ;
```

Result Grid

company	iban	average_transaction
Donec Ltd	PT87806228135092429456346	203.715000

Result 7 x

Output

Action Output

#	Time	Action	Message
✓ 1	10:27:11	SELECT companies.company_name as company , credit_cards.iban AS iban , AVG(tra...	1 row(s) returned

Nivel 2

Crear una nueva tabla que refleje el estado de las tarjetas de crédito en función de si las tres últimas transacciones fueron rechazadas y generar la siguiente consulta:

Ejercicio 1

¿Cuántas tarjetas están activas?

Se crea una nueva tabla denominada `credit_cards_status`.

Se insertan los datos `card_id` y el valor 'active' o 'non-active' en la columna "status" según las condiciones establecidas.

Para la asignación de la condición de activa o no activa se ha realizado un filtrado de las tres ultimas transacciones usando la función `dense rank` y una tabla CTE y la condición de rechazo en función de si la suma de la columna 'declined' en las tres ultimas transacciones es menor de tres o no.

La columna 'declined' asigna un 1 para las transacciones rechazadas y un 0 para las transacciones aceptadas.

Creación de la tabla credit_cards_status

Se crea la tabla credit_card_status.

El tipo de datos es definido como VARCHAR en todos los campos.

El campo "status" solo tiene dos valores: 'active' o 'non-active'.

El campo "card_id" es asignado como PRIMARY KEY.

```
175 -- Creacion de la tabla credit_card_status.
176 • CREATE TABLE credit_cards_status
177 (
178     card_id VARCHAR(255) PRIMARY KEY
179     , status VARCHAR(255)
180 )
181 ;
```

Comprobación de la tabla credit_cards_status.

```
183 -- Comprobacion de la tabla credit_cards_status.
184 • SHOW COLUMNS
185 from credit_cards_status
186 ;
187
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [\[A\]](#)

	Field	Type	Null	Key	Default	Extra
▶	card_id	varchar(255)	NO	PRI	NULL	
	status	varchar(255)	YES		NULL	

Result 17 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:07:29	SHOW COLUMNS from credit_cards_status	2 row(s) returned
✓ 2	12:08:20	SHOW COLUMNS from credit_cards_status	2 row(s) returned

Introducción de datos en la tabla credit_cards_status.

```
188 -- Introduccion de datos en la tabla credit_cards_status.
189 • INSERT INTO credit_cards_status
190     WITH rank_num_transactions AS
191     (
192     SELECT card_id
193     , timestamp
194     , declined
195     , DENSE_RANK() OVER ( PARTITION BY card_id
196                           ORDER BY timestamp DESC
197                           ) AS num_trans
198     FROM transactions
199     )
200
201     SELECT card_id
202     , CASE
203         WHEN SUM(declined) < 3
204         THEN 'active'
205         ELSE 'non-active'
206     END AS status
207     FROM rank_num_transactions
208     WHERE num_trans <= 3
209     GROUP BY card_id
210 ;
```

Output

Action Output

#	Time	Action	Message
✓ 1	11:12:15	INSERT INTO credit_cards_status WITH rank_num_transactions AS (SELECT card_id ...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

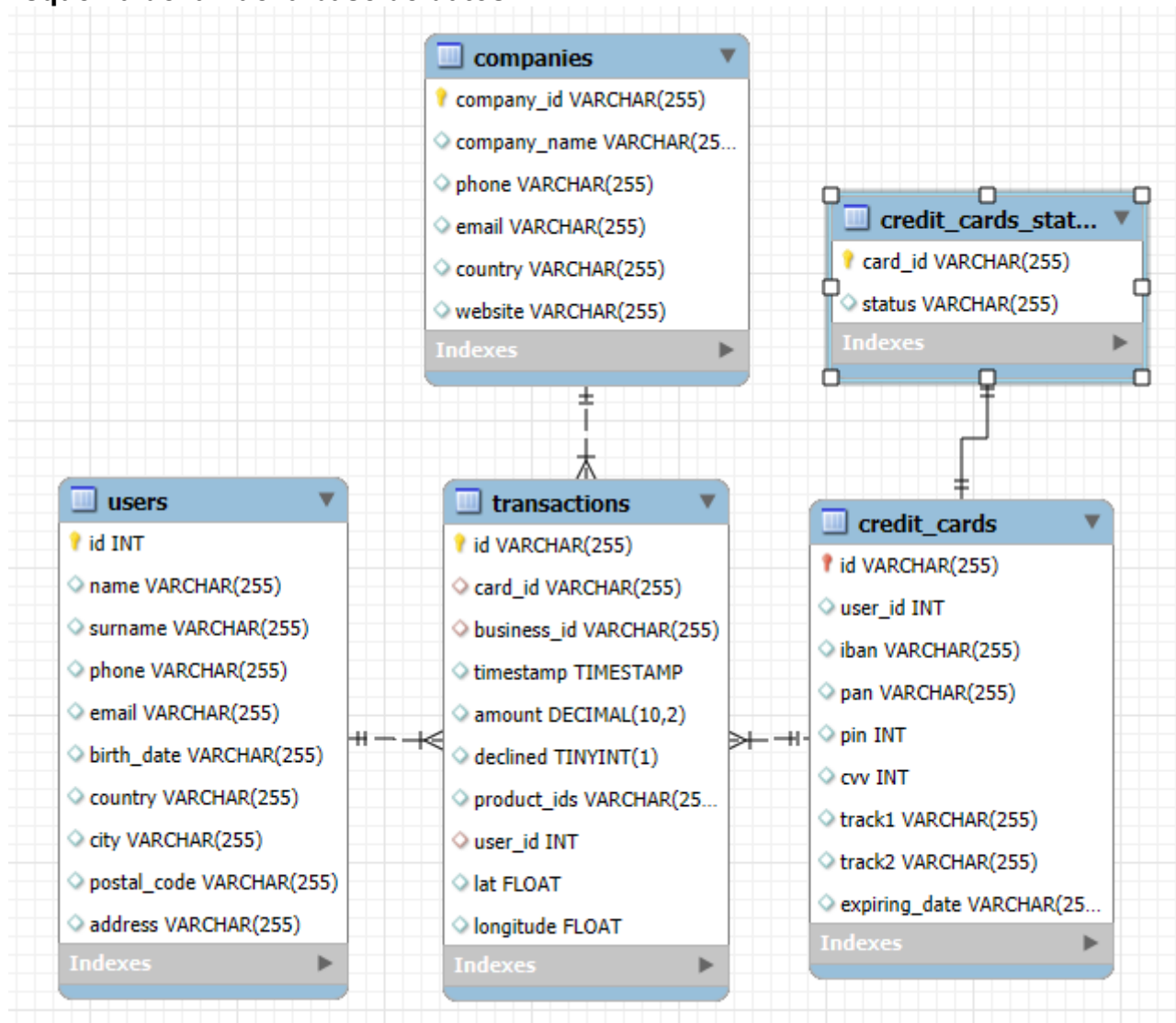
Creación de la relación entre credit_cards y credit_cards_status.

Se crea una foreign key en la tabla credit_cards referenciada a la tabla credit_cards_status.

Se crea una relación 1:1 entre las tablas credit_cards y credit_cards_status.

```
211
212 -- Establecer la relacion entre credit_cards y credit_cards_status
213 • ALTER TABLE credit_cards
214     ADD FOREIGN KEY credit_cards(id) REFERENCES credit_cards_status(card_id)
215 ;
```

Esquema de la nueva base de datos.



Nivel 2

Ejercicio 1

¿Cuántas tarjetas están activas?

Número de tarjetas activas.

```
212 -- Nivel 2
213 -- Ejercicio 1
214 /*
215 Ejercicio 1
216 ¿Cuántas tarjetas están activas?
217 */
218 • SELECT COUNT(card_id) AS num_cards_active
219 FROM credit_cards_status
220 WHERE status = 'active'
221 ;
222
```

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

	num_cards_active
▶	275

Result 13 x

Output



Action Output



	#	Time	Action	Message
✓	1	11:19:17	SELECT COUNT(card_id) AS num_cards_active FROM credit_cards_status WHERE s...	1 row(s) returned

Nivel 3

Crea una tabla con la cual se puedan unir los datos del nuevo archivo productos.csv con la base de datos creada, teniendo en cuenta que desde la tabla transactions tienes product_ids. Genera la siguiente consulta:

Exercici 1

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

Creación de la tabla productos.

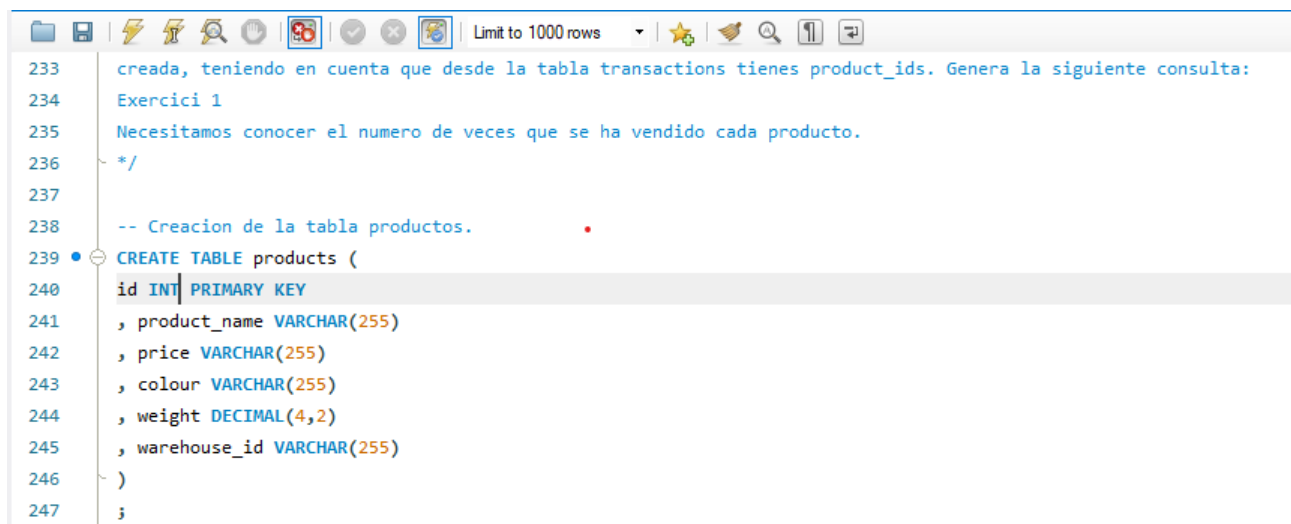
Se crea la tabla productos.

Los campos tipo texto se definen como VARCHAR.

El campo "id" se define como INT.

El campo "price" se define como VARCHAR ya que incluye el simbolo de la unidad monetaria usada: '\$'. Para realizar cálculos en función de este campo habría que extraer la parte numerica del campo y convertirla a tipo INT.

El campo "weight" se define como DECIMAL.



```
233 creada, teniendo en cuenta que desde la tabla transactions tienes product_ids. Genera la siguiente consulta:
234 Exercici 1
235 Necesitamos conocer el numero de veces que se ha vendido cada producto.
236 */
237
238 -- Creacion de la tabla productos.
239 CREATE TABLE products (
240   id INT PRIMARY KEY
241   , product_name VARCHAR(255)
242   , price VARCHAR(255)
243   , colour VARCHAR(255)
244   , weight DECIMAL(4,2)
245   , warehouse_id VARCHAR(255)
246 )
247 ;
```

Comprobación de la tabla productos.

Limit to 1000 rows

```
248 -- Comprobacion de la tabla productos.
249 SHOW COLUMNS
250 FROM products
251 ;
252
```

Result Grid

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
product_name	varchar(255)	YES		NULL	
price	varchar(255)	YES		NULL	
colour	varchar(255)	YES		NULL	
weight	decimal(4,2)	YES		NULL	
warehouse_id	varchar(255)	YES		NULL	

Result 21

Output

Action Output

#	Time	Action	Message
1	13:05:23	SHOW COLUMNS FROM products	6 row(s) returned

Insertar datos desde producto.csv en la tabla products.

Limit to 1000 rows

```
250 ;
251
252 -- Introduccion de datos desde products.csv en la tabla products.
253 LOAD DATA LOCAL INFILE "C:/Descargas/products.csv"
254 INTO TABLE products
255 FIELDS TERMINATED BY ','
256 ENCLOSED BY '"'
257 LINES TERMINATED BY '\n'
258 IGNORE 1 ROWS
259 ;
260
```

Output

Action Output

#	Time	Action	Message
1	12:45:47	LOAD DATA LOCAL INFILE "C:/Users/MAITE/Desktop/JOSE/IT_ACADEMY/2_Espe...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Creación de la tabla puente transactions_products.

Se crea la tabla puente transactions_products.

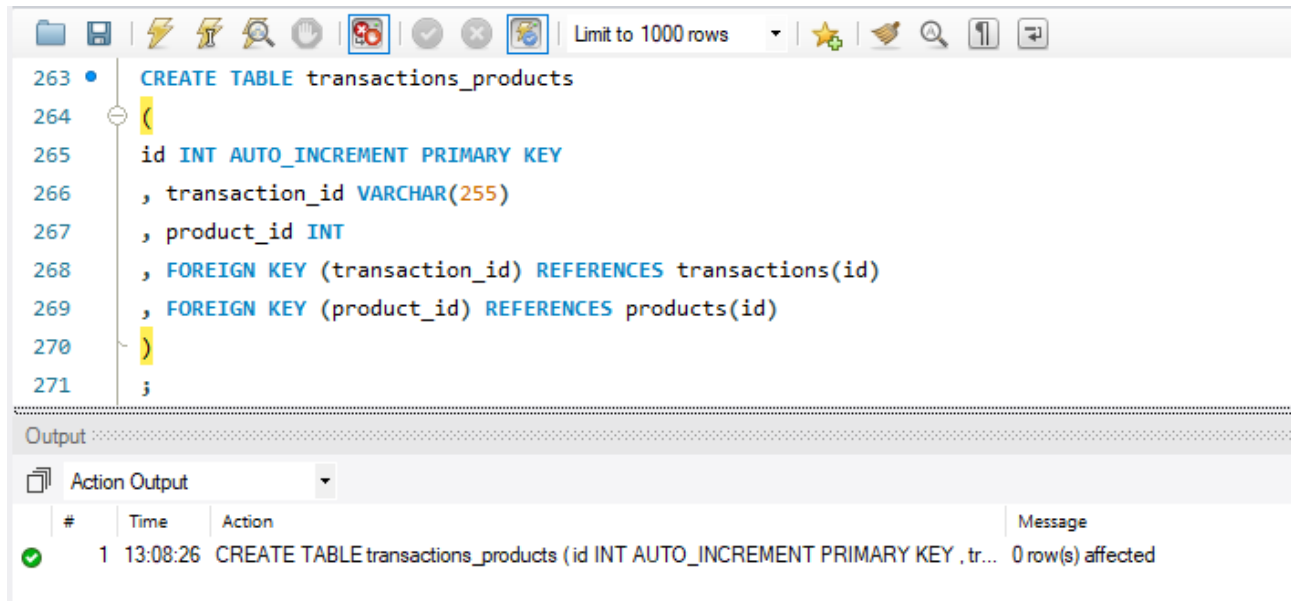
Se crea un campo "id" de tipo INT autoincremental. Este campo se define como PRIMARY KEY.

Se crea el campo "transaction_id" definido como VARCHAR. Este campo es asignado como FOREIGN KEY referenciado al campo "id" de la tabla transactions.

Se crea el campo "product_id" definido como INT. Es asignado como FOREIGN KEY referenciado al al campo "id" de la tabla products.

Se establece una relación 1:N entre las tablas transactions y transactions_products.

Se establece una relación 1:N entre las tablas products y transactions_products.



The screenshot shows a database management interface with a SQL editor and an output window. The SQL editor contains the following code:

```
263 • CREATE TABLE transactions_products
264 (
265     id INT AUTO_INCREMENT PRIMARY KEY
266     , transaction_id VARCHAR(255)
267     , product_id INT
268     , FOREIGN KEY (transaction_id) REFERENCES transactions(id)
269     , FOREIGN KEY (product_id) REFERENCES products(id)
270 )
271 ;
```

The output window shows the result of the SQL execution:

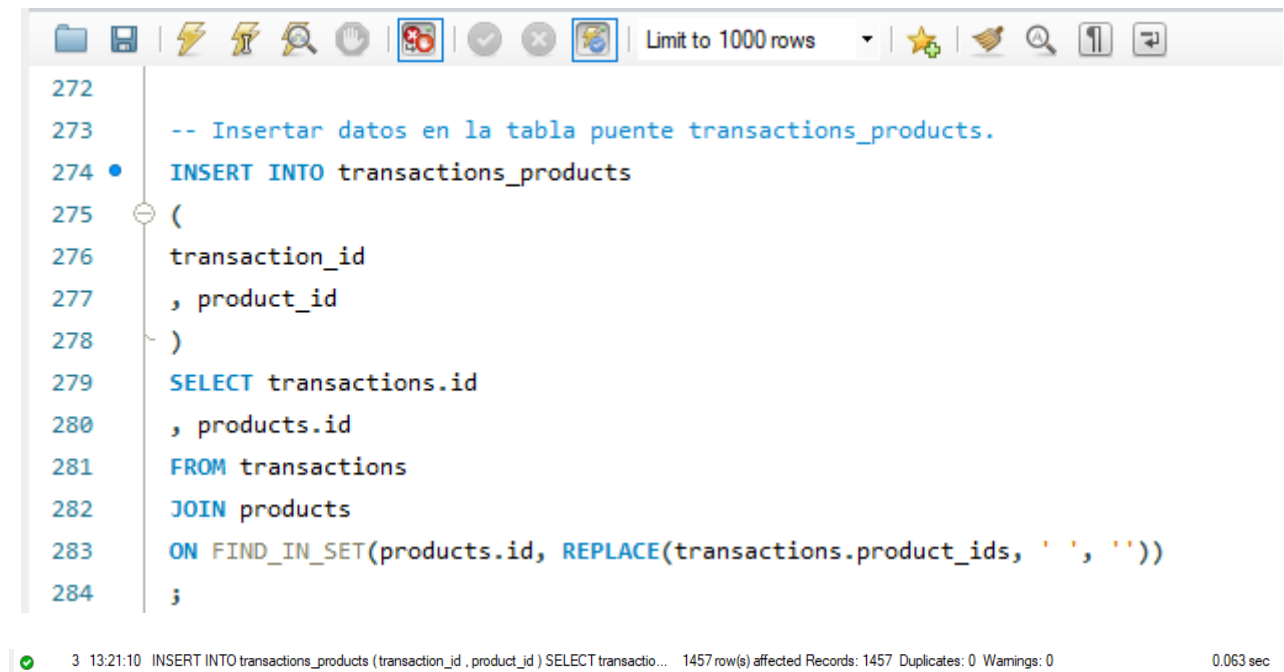
#	Time	Action	Message
✓ 1	13:08:26	CREATE TABLE transactions_products (id INT AUTO_INCREMENT PRIMARY KEY , tr...	0 row(s) affected

Insertar datos en la tabla puente transactions_products.

Se introducen en la tabla puente el id de cada transacción y el id de cada producto vendido en cada transacción en filas independientes.

Se seleccionan los id de cada producto dentro del campo transactions.product_ids que existen en la tabla productos por medio del comando FIND_IN_SET.

Se eliminan los espacios existente en la columna product_ids de la tabla transactions dado que el comando find_in_set trabaja con datos separados por comas sin espacios.

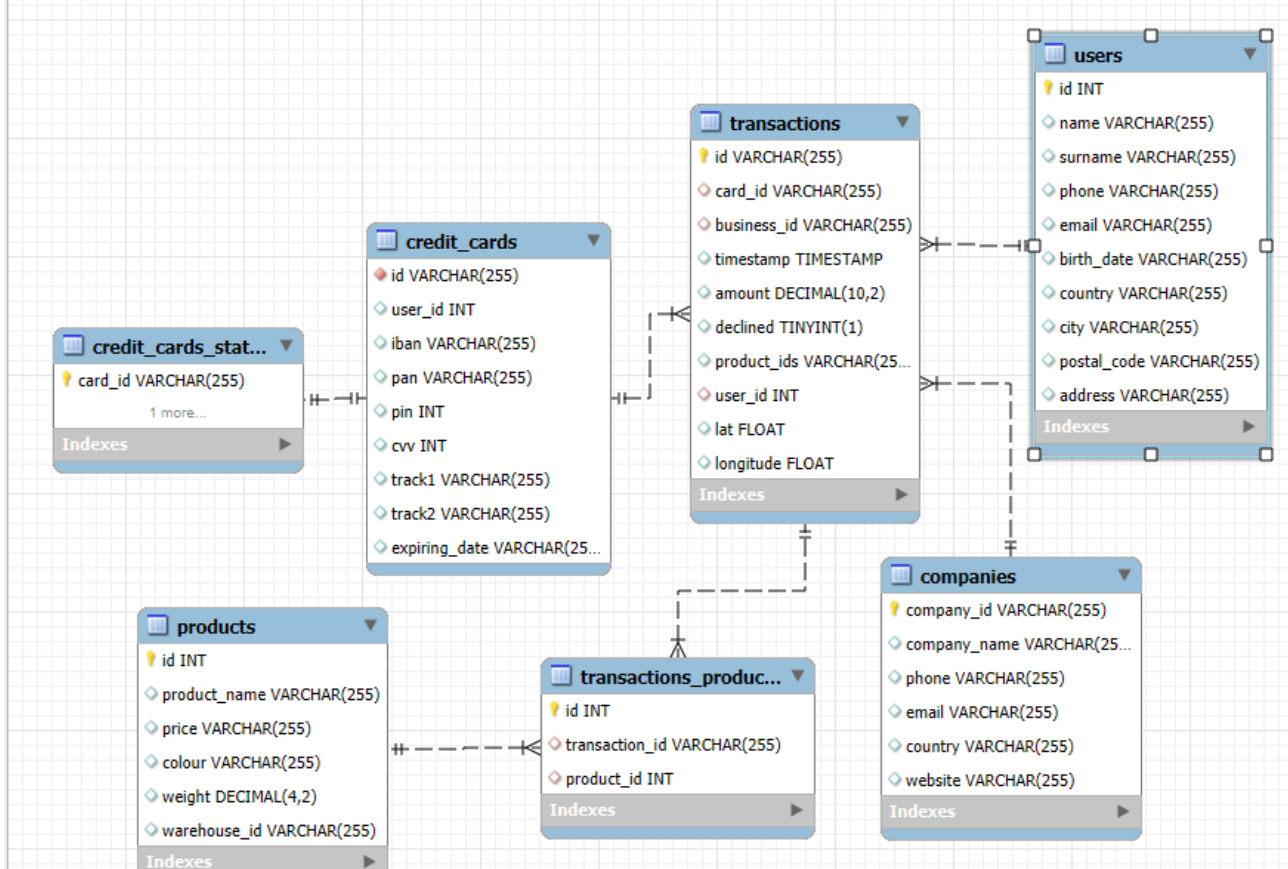


```
272
273  -- Insertar datos en la tabla puente transactions_products.
274  INSERT INTO transactions_products
275  (
276    transaction_id
277    , product_id
278  )
279  SELECT transactions.id
280    , products.id
281  FROM transactions
282  JOIN products
283  ON FIND_IN_SET(products.id, REPLACE(transactions.product_ids, ' ', ''))
284  ;
```

3 13:21:10 INSERT INTO transactions_products (transaction_id , product_id) SELECT transactio... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0 0.063 sec

Esquema de la base de datos.

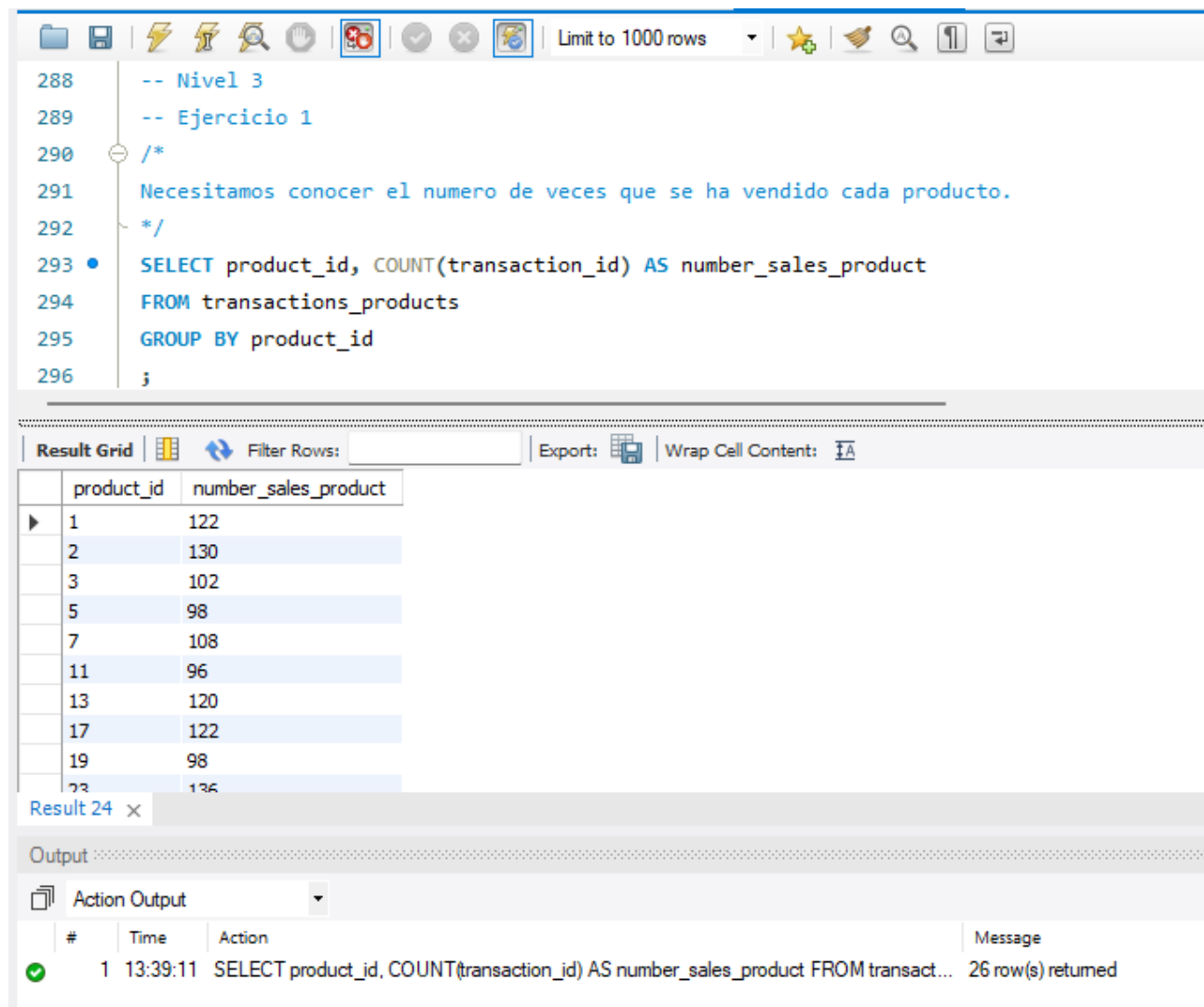
El diseño final de la base de datos corresponde a un modelo en copo de nieve.



Nivel 3

Ejercicio 1

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



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and a 'Limit to 1000 rows' dropdown. The SQL editor contains the following code:

```
288 -- Nivel 3
289 -- Ejercicio 1
290 /*
291 Necesitamos conocer el numero de veces que se ha vendido cada producto.
292 */
293 • SELECT product_id, COUNT(transaction_id) AS number_sales_product
294 FROM transactions_products
295 GROUP BY product_id
296 ;
```

Below the editor is the 'Result Grid' tab, which displays the query results in a table:

	product_id	number_sales_product
▶	1	122
	2	130
	3	102
	5	98
	7	108
	11	96
	13	120
	17	122
	19	98
	23	136

Below the result grid is the 'Output' tab, which shows the execution log:

#	Time	Action	Message
✓ 1	13:39:11	SELECT product_id, COUNT(transaction_id) AS number_sales_product FROM transact...	26 row(s) returned