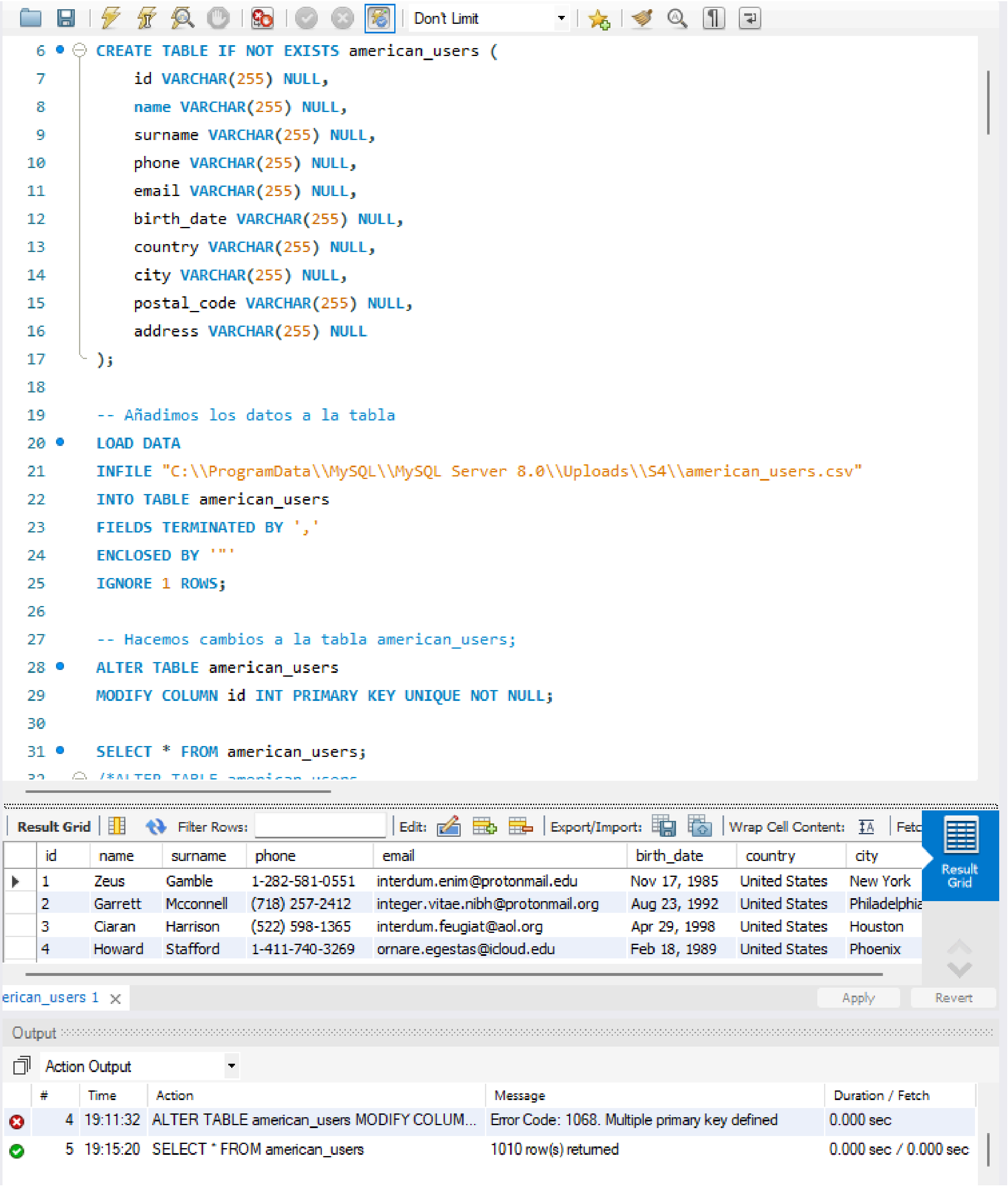
Tasca S4.01. Creació de Base de Dades

Nivell 1

Exercici 1

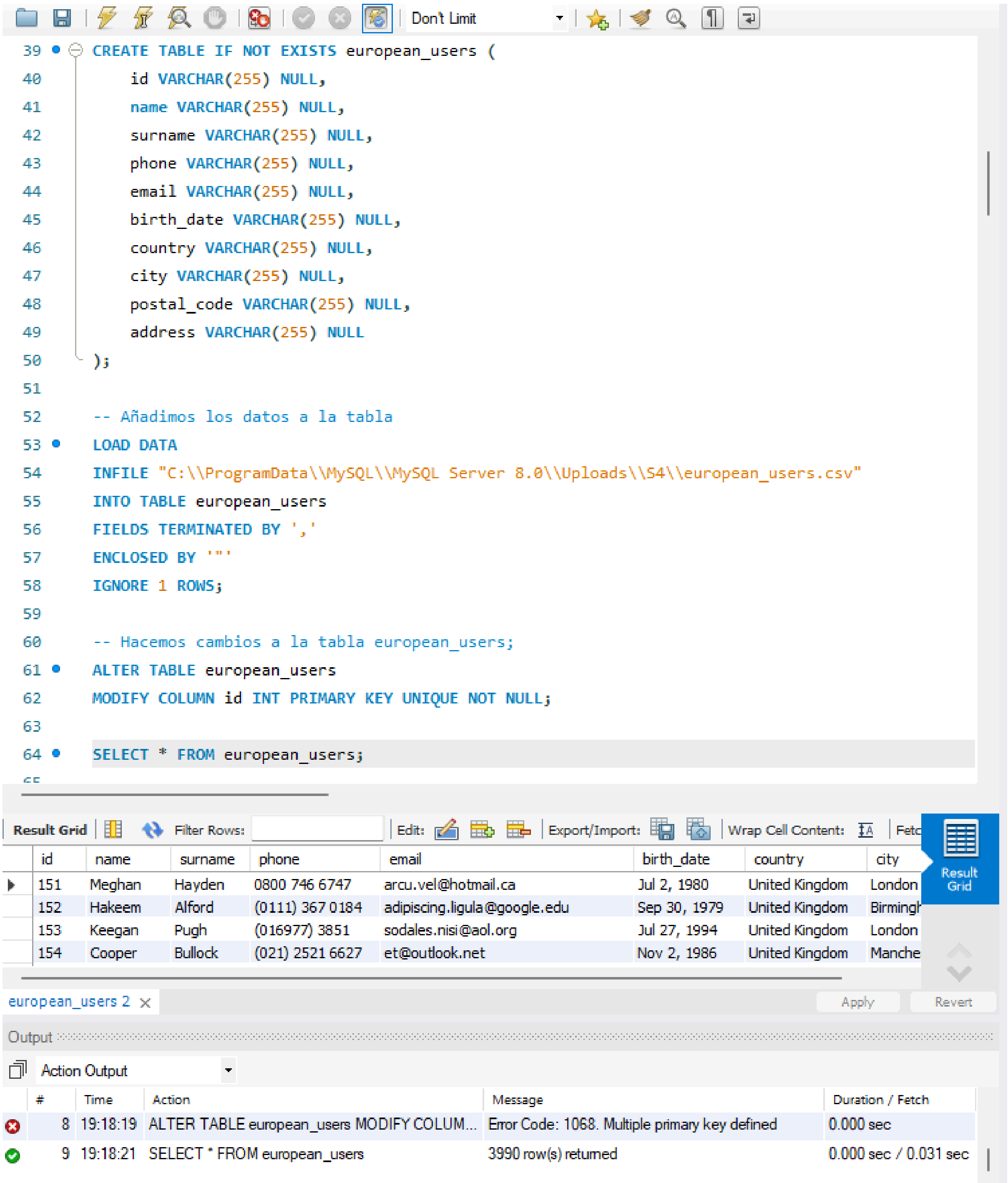
En esta tarea, se decidió añadir a la base de datos todos los archivos CSV excepto product.csv. Todas las tablas creadas tenían las columnas iniciales de tipo VARCHAR, para evitar problemas en la lectura, y posteriormente se modificaron los tipos de valores de las columnas dependiendo de la tabla. También se añadieron las primary keys y foreign keys más adelante. Las primeras tablas creadas fueron american_users y european_users, que posteriormente se combinaron en la tabla users.



```sql
CREATE TABLE IF NOT EXISTS american_users (
    id VARCHAR(255) NULL,
    name VARCHAR(255) NULL,
    surname VARCHAR(255) NULL,
    phone VARCHAR(255) NULL,
    email VARCHAR(255) NULL,
    birth_date VARCHAR(255) NULL,
    country VARCHAR(255) NULL,
    city VARCHAR(255) NULL,
    postal_code VARCHAR(255) NULL,
    address VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\american_users.csv"
INTO TABLE american_users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla american_users;
ALTER TABLE american_users
MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;

SELECT * FROM american_users;
/*ALTER TABLE american_users
```

	id	name	surname	phone	email	birth_date	country	city
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia
	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix

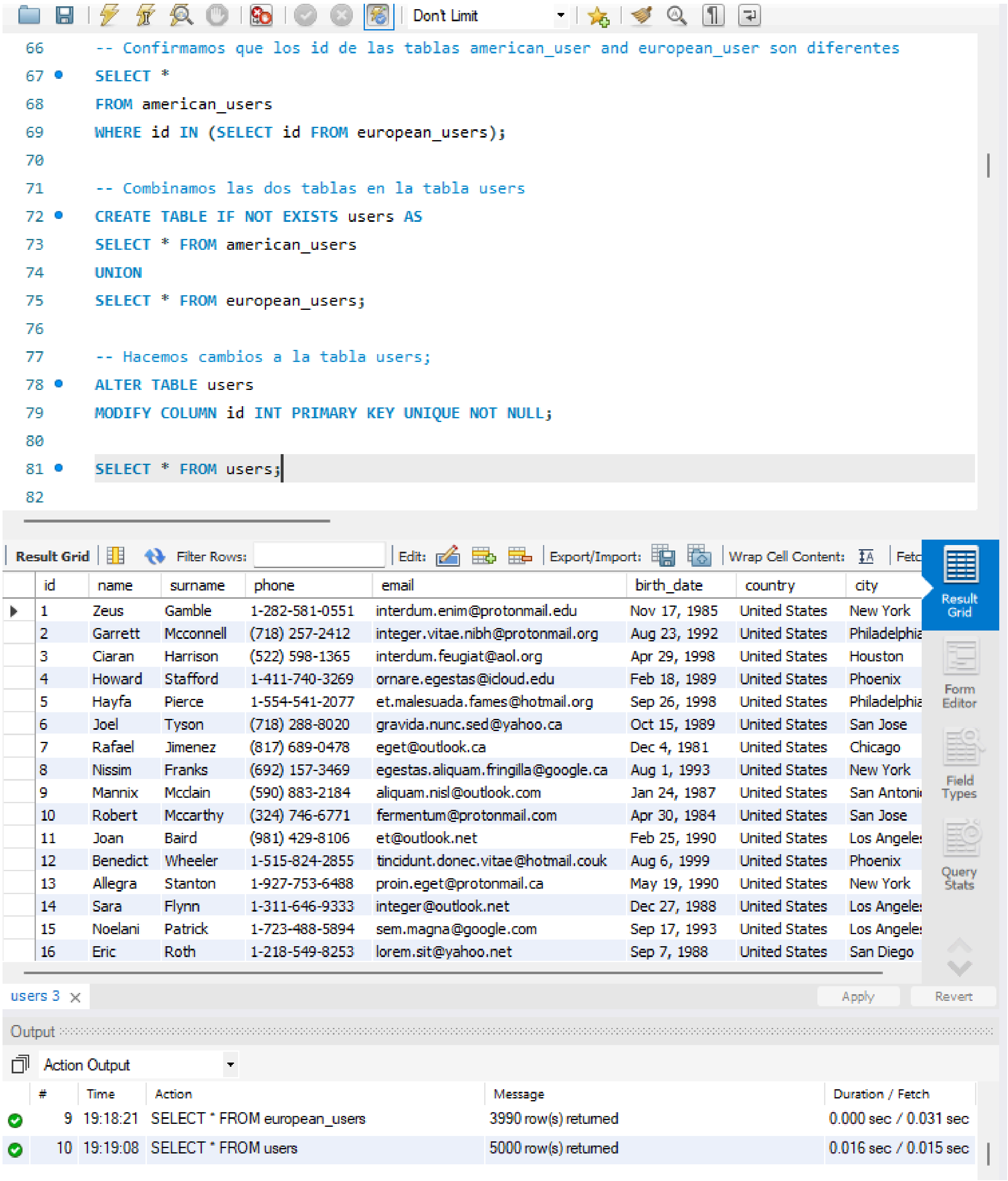
	#	Time	Action	Message	Duration / Fetch
✖	4	19:11:32	ALTER TABLE american_users MODIFY COLUM...	Error Code: 1068. Multiple primary key defined	0.000 sec
✓	5	19:15:20	SELECT * FROM american_users	1010 row(s) returned	0.000 sec / 0.000 sec



```sql
CREATE TABLE IF NOT EXISTS european_users (
    id VARCHAR(255) NULL,
    name VARCHAR(255) NULL,
    surname VARCHAR(255) NULL,
    phone VARCHAR(255) NULL,
    email VARCHAR(255) NULL,
    birth_date VARCHAR(255) NULL,
    country VARCHAR(255) NULL,
    city VARCHAR(255) NULL,
    postal_code VARCHAR(255) NULL,
    address VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\european_users.csv"
INTO TABLE european_users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla european_users;
ALTER TABLE european_users
MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;

SELECT * FROM european_users;
```

id	name	surname	phone	email	birth_date	country	city
151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	Jul 2, 1980	United Kingdom	London
152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google.edu	Sep 30, 1979	United Kingdom	Birmingh
153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	London
154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Manche

#	Time	Action	Message	Duration / Fetch
8	19:18:19	ALTER TABLE european_users MODIFY COLUM...	Error Code: 1068. Multiple primary key defined	0.000 sec
9	19:18:21	SELECT * FROM european_users	3990 row(s) returned	0.000 sec / 0.031 sec

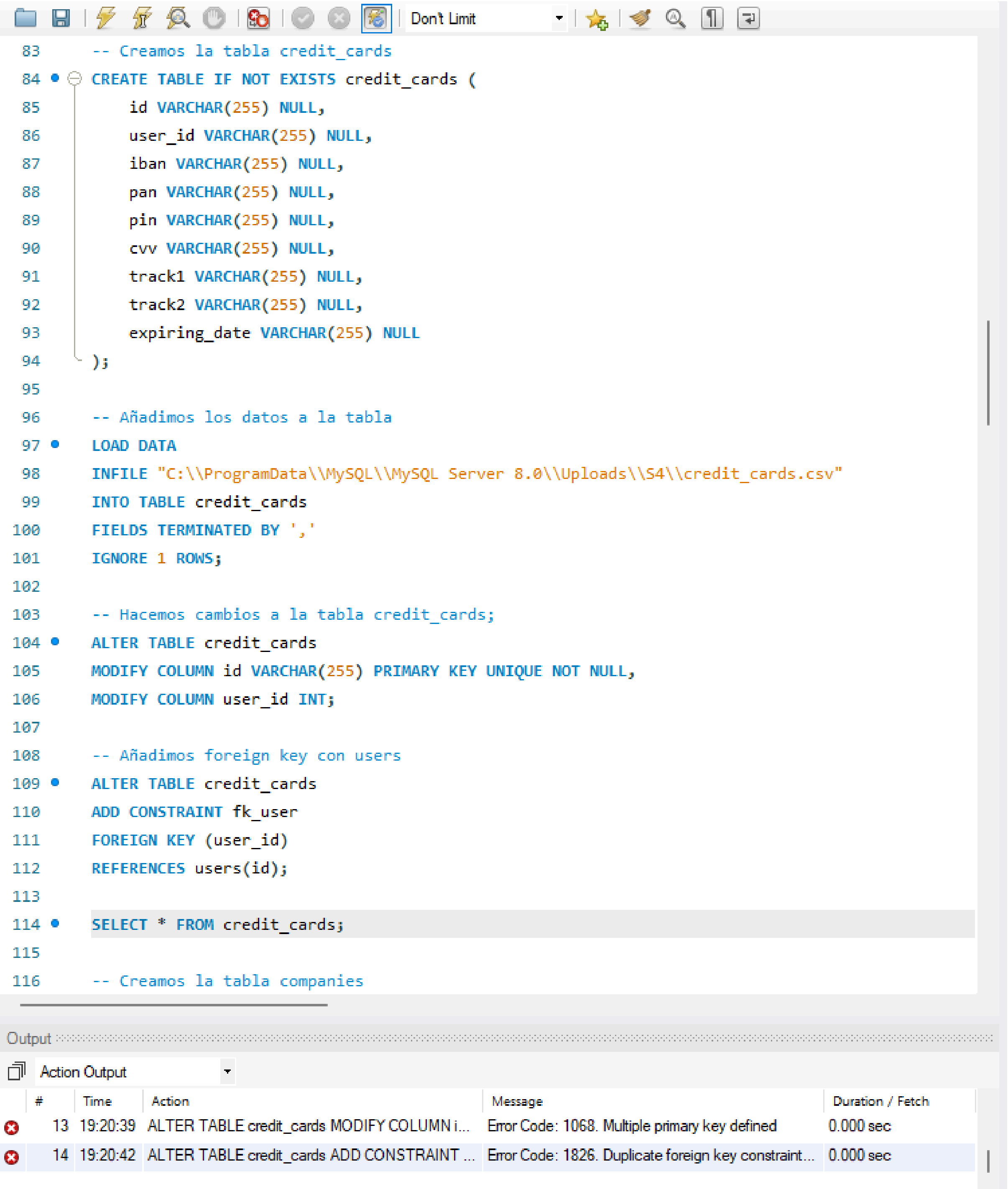


```sql
-- Confirmamos que los id de las tablas american_user and european_user son diferentes
SELECT *
FROM american_users
WHERE id IN (SELECT id FROM european_users);

-- Combinamos las dos tablas en la tabla users
CREATE TABLE IF NOT EXISTS users AS
SELECT * FROM american_users
UNION
SELECT * FROM european_users;

-- Hacemos cambios a la tabla users;
ALTER TABLE users
MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;

SELECT * FROM users;
```

id	name	surname	phone	email	birth_date	country	city
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Philadelphia
6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	San Jose
7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Chicago
8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	New York
9	Mannix	Mcclain	(590) 883-2184	aliquam.nisl@outlook.com	Jan 24, 1987	United States	San Antoni
10	Robert	Mccarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	San Jose
11	Joan	Baird	(981) 429-8106	et@outlook.net	Feb 25, 1990	United States	Los Angele
12	Benedict	Wheeler	1-515-824-2855	tincidunt.donec.vitae@hotmail.couk	Aug 6, 1999	United States	Phoenix
13	Allegra	Stanton	1-927-753-6488	proin.eget@protonmail.ca	May 19, 1990	United States	New York
14	Sara	Flynn	1-311-646-9333	integer@outlook.net	Dec 27, 1988	United States	Los Angele
15	Noelani	Patrick	1-723-488-5894	sem.magna@google.com	Sep 17, 1993	United States	Los Angele
16	Eric	Roth	1-218-549-8253	lorem.sit@yahoo.net	Sep 7, 1988	United States	San Diego

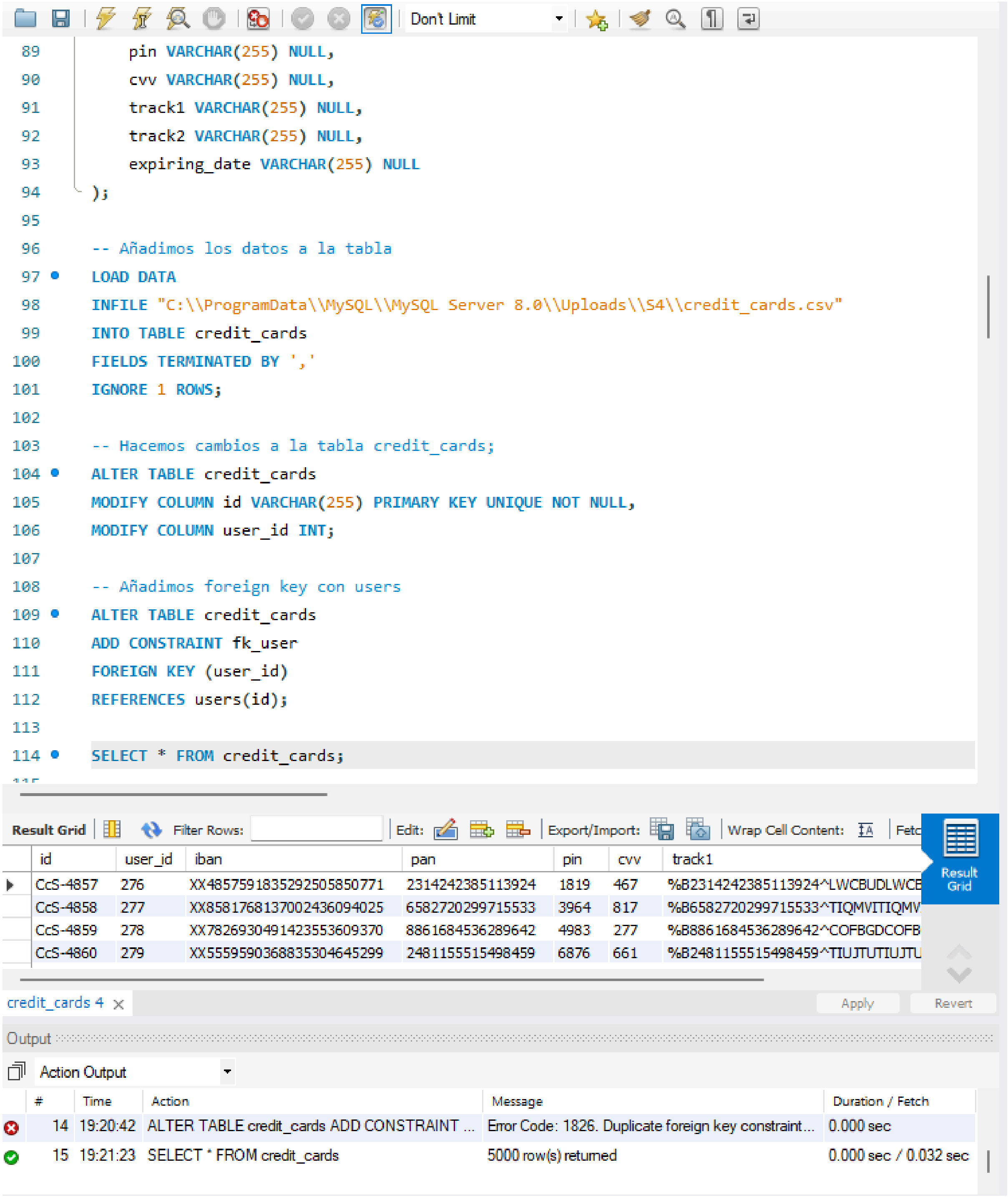
	#	Time	Action	Message	Duration / Fetch
✓	9	19:18:21	SELECT * FROM european_users	3990 row(s) returned	0.000 sec / 0.031 sec
✓	10	19:19:08	SELECT * FROM users	5000 row(s) returned	0.016 sec / 0.015 sec

Añadimos el resto de las tablas: credit_cards, companies y transactions.



```sql
-- Creamos la tabla credit_cards
CREATE TABLE IF NOT EXISTS credit_cards (
    id VARCHAR(255) NULL,
    user_id VARCHAR(255) NULL,
    iban VARCHAR(255) NULL,
    pan VARCHAR(255) NULL,
    pin VARCHAR(255) NULL,
    cvv VARCHAR(255) NULL,
    track1 VARCHAR(255) NULL,
    track2 VARCHAR(255) NULL,
    expiring_date VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\credit_cards.csv"
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla credit_cards;
ALTER TABLE credit_cards
MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
MODIFY COLUMN user_id INT;

-- Añadimos foreign key con users
ALTER TABLE credit_cards
ADD CONSTRAINT fk_user
FOREIGN KEY (user_id)
REFERENCES users(id);

SELECT * FROM credit_cards;

-- Creamos la tabla companies
```

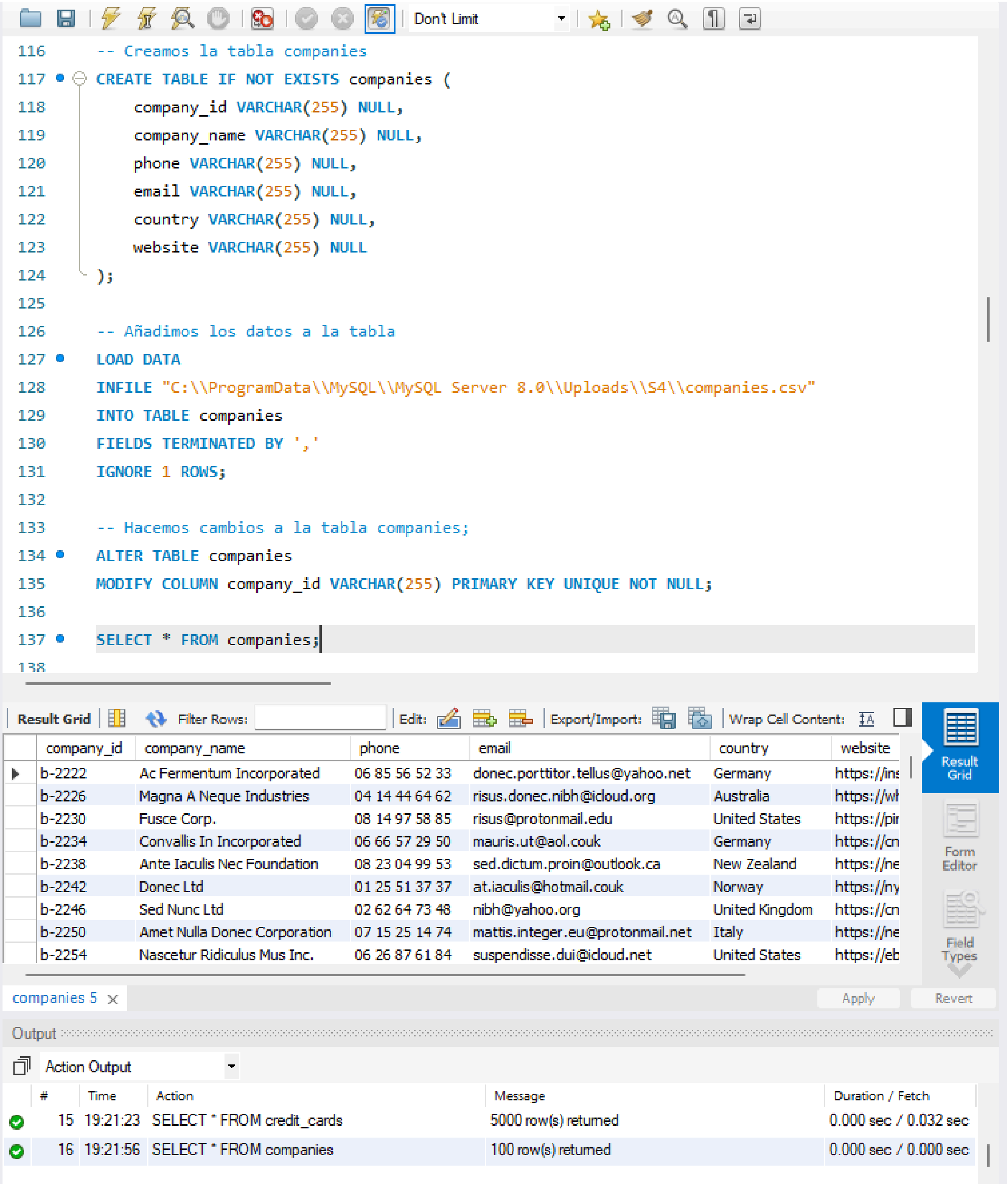
	#	Time	Action	Message	Duration / Fetch
☒	13	19:20:39	ALTER TABLE credit_cards MODIFY COLUMN i...	Error Code: 1068. Multiple primary key defined	0.000 sec
☒	14	19:20:42	ALTER TABLE credit_cards ADD CONSTRAINT ...	Error Code: 1826. Duplicate foreign key constraint...	0.000 sec



```sql
    pin VARCHAR(255) NULL,
    cvv VARCHAR(255) NULL,
    track1 VARCHAR(255) NULL,
    track2 VARCHAR(255) NULL,
    expiring_date VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\credit_cards.csv"
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla credit_cards;
ALTER TABLE credit_cards
MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
MODIFY COLUMN user_id INT;

-- Añadimos foreign key con users
ALTER TABLE credit_cards
ADD CONSTRAINT fk_user
FOREIGN KEY (user_id)
REFERENCES users(id);

SELECT * FROM credit_cards;
```

id	user_id	iban	pan	pin	cvv	track1
CcS-4857	276	XX4857591835292505850771	2314242385113924	1819	467	%B2314242385113924^LWCBUDLWCB
CcS-4858	277	XX8581768137002436094025	6582720299715533	3964	817	%B6582720299715533^TIQMVITIQMV
CcS-4859	278	XX7826930491423553609370	8861684536289642	4983	277	%B8861684536289642^COFBGDCOFB
CcS-4860	279	XX5559590368835304645299	2481155515498459	6876	661	%B2481155515498459^TIUJTUTIUJTU

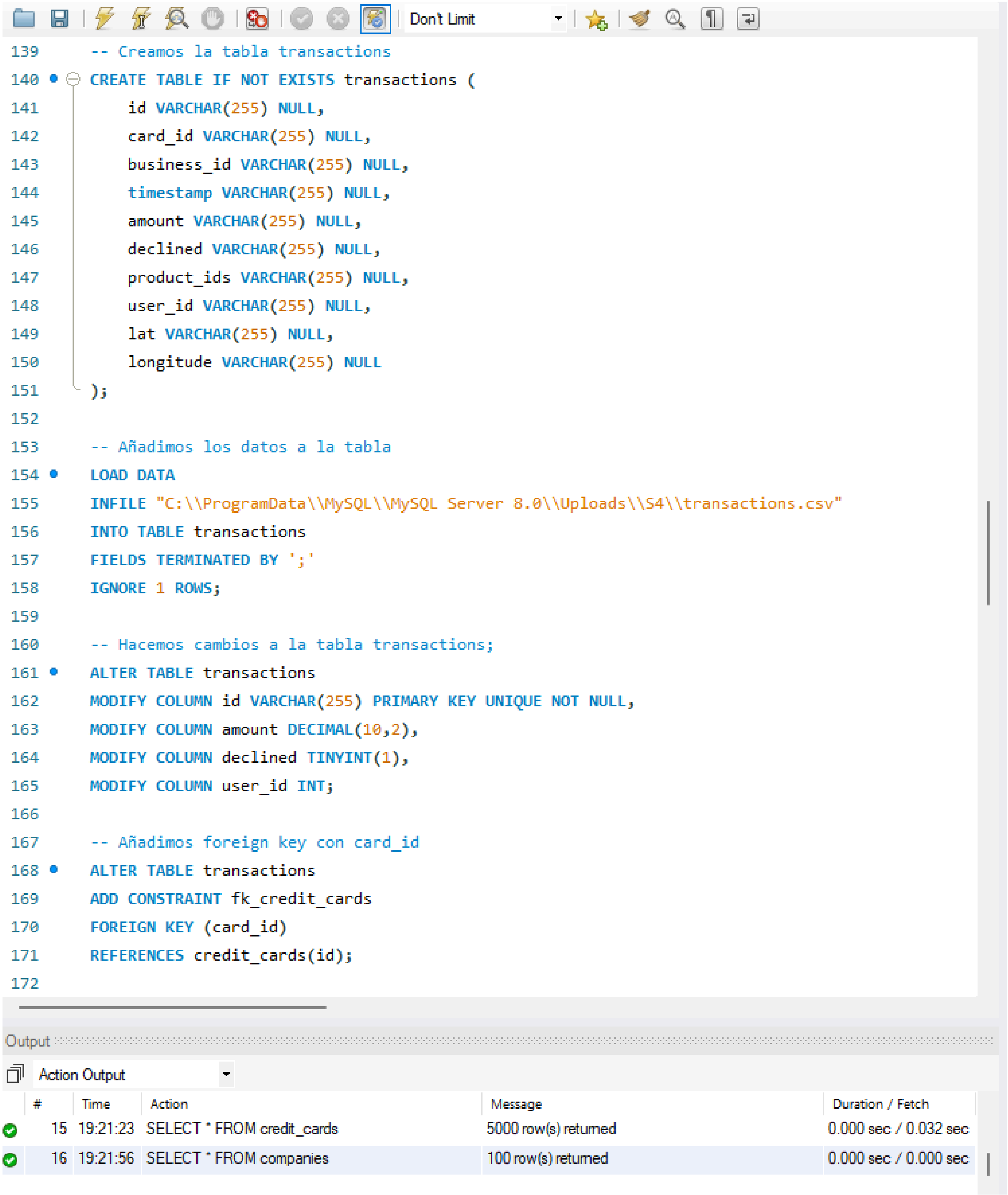
	#	Time	Action	Message	Duration / Fetch
✗	14	19:20:42	ALTER TABLE credit_cards ADD CONSTRAINT ...	Error Code: 1826. Duplicate foreign key constraint...	0.000 sec
✓	15	19:21:23	SELECT * FROM credit_cards	5000 row(s) returned	0.000 sec / 0.032 sec



```sql
-- Creamos la tabla companies
CREATE TABLE IF NOT EXISTS companies (
    company_id VARCHAR(255) NULL,
    company_name VARCHAR(255) NULL,
    phone VARCHAR(255) NULL,
    email VARCHAR(255) NULL,
    country VARCHAR(255) NULL,
    website VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\companies.csv"
INTO TABLE companies
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla companies;
ALTER TABLE companies
MODIFY COLUMN company_id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL;

SELECT * FROM companies;
```

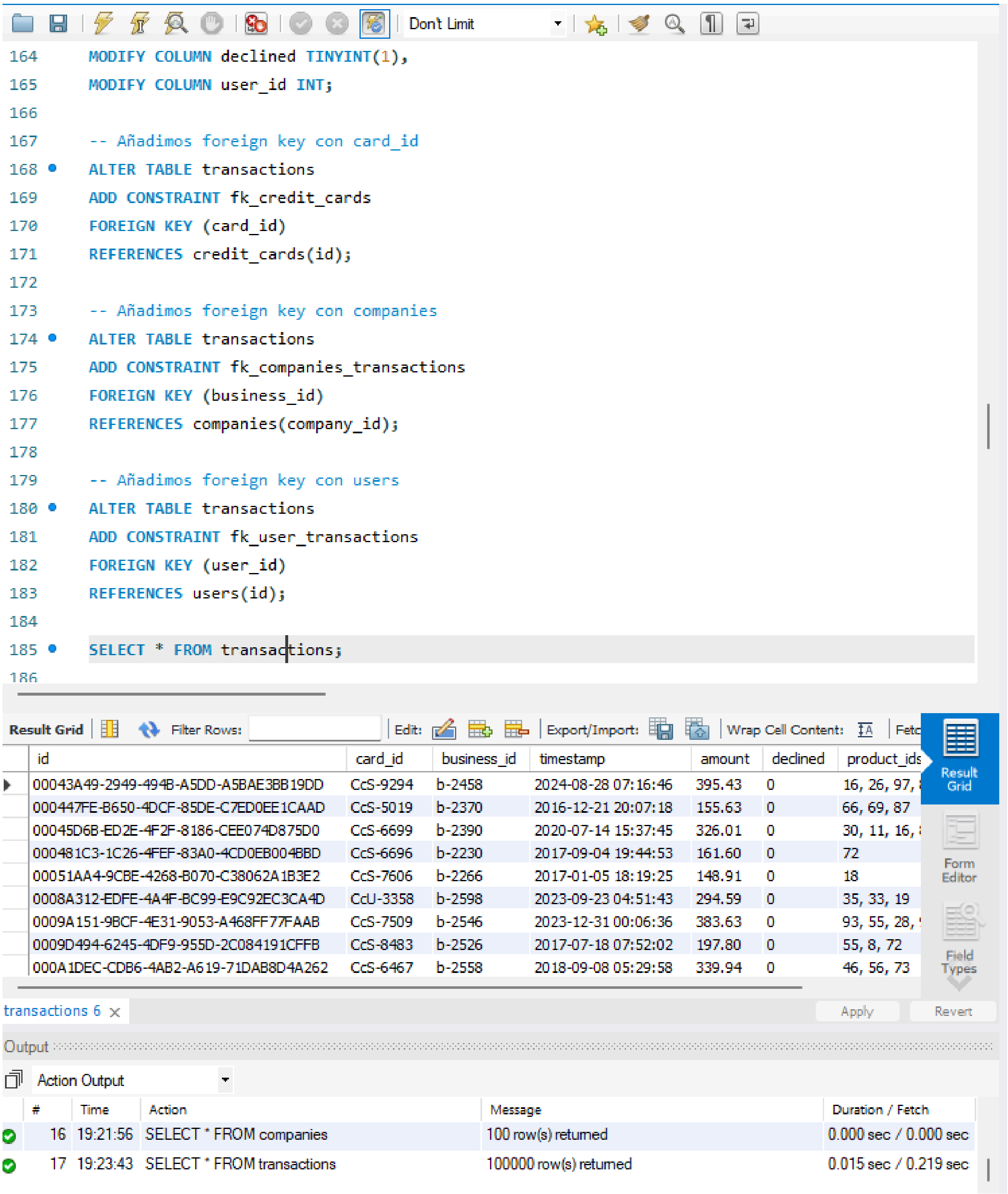
company_id	company_name	phone	email	country	website
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://ins
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://wh
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pir
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany	https://cn
b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.proin@outlook.ca	New Zealand	https://ne
b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.couk	Norway	https://ny
b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cn
b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://ne
b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@icloud.net	United States	https://eb

#	Time	Action	Message	Duration / Fetch
15	19:21:23	SELECT * FROM credit_cards	5000 row(s) returned	0.000 sec / 0.032 sec
16	19:21:56	SELECT * FROM companies	100 row(s) returned	0.000 sec / 0.000 sec



```
-- Creamos la tabla transactions
CREATE TABLE IF NOT EXISTS transactions (
    id VARCHAR(255) NULL,
    card_id VARCHAR(255) NULL,
    business_id VARCHAR(255) NULL,
    timestamp VARCHAR(255) NULL,
    amount VARCHAR(255) NULL,
    declined VARCHAR(255) NULL,
    product_ids VARCHAR(255) NULL,
    user_id VARCHAR(255) NULL,
    lat VARCHAR(255) NULL,
    longitude VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\transactions.csv"
INTO TABLE transactions
FIELDS TERMINATED BY ';'
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla transactions;
ALTER TABLE transactions
MODIFY COLUMN id VARCHAR(255) PRIMARY KEY UNIQUE NOT NULL,
MODIFY COLUMN amount DECIMAL(10,2),
MODIFY COLUMN declined TINYINT(1),
MODIFY COLUMN user_id INT;

-- Añadimos foreign key con card_id
ALTER TABLE transactions
ADD CONSTRAINT fk_credit_cards
FOREIGN KEY (card_id)
REFERENCES credit_cards(id);
```

	#	Time	Action	Message	Duration / Fetch
✓	15	19:21:23	SELECT * FROM credit_cards	5000 row(s) returned	0.000 sec / 0.032 sec
✓	16	19:21:56	SELECT * FROM companies	100 row(s) returned	0.000 sec / 0.000 sec

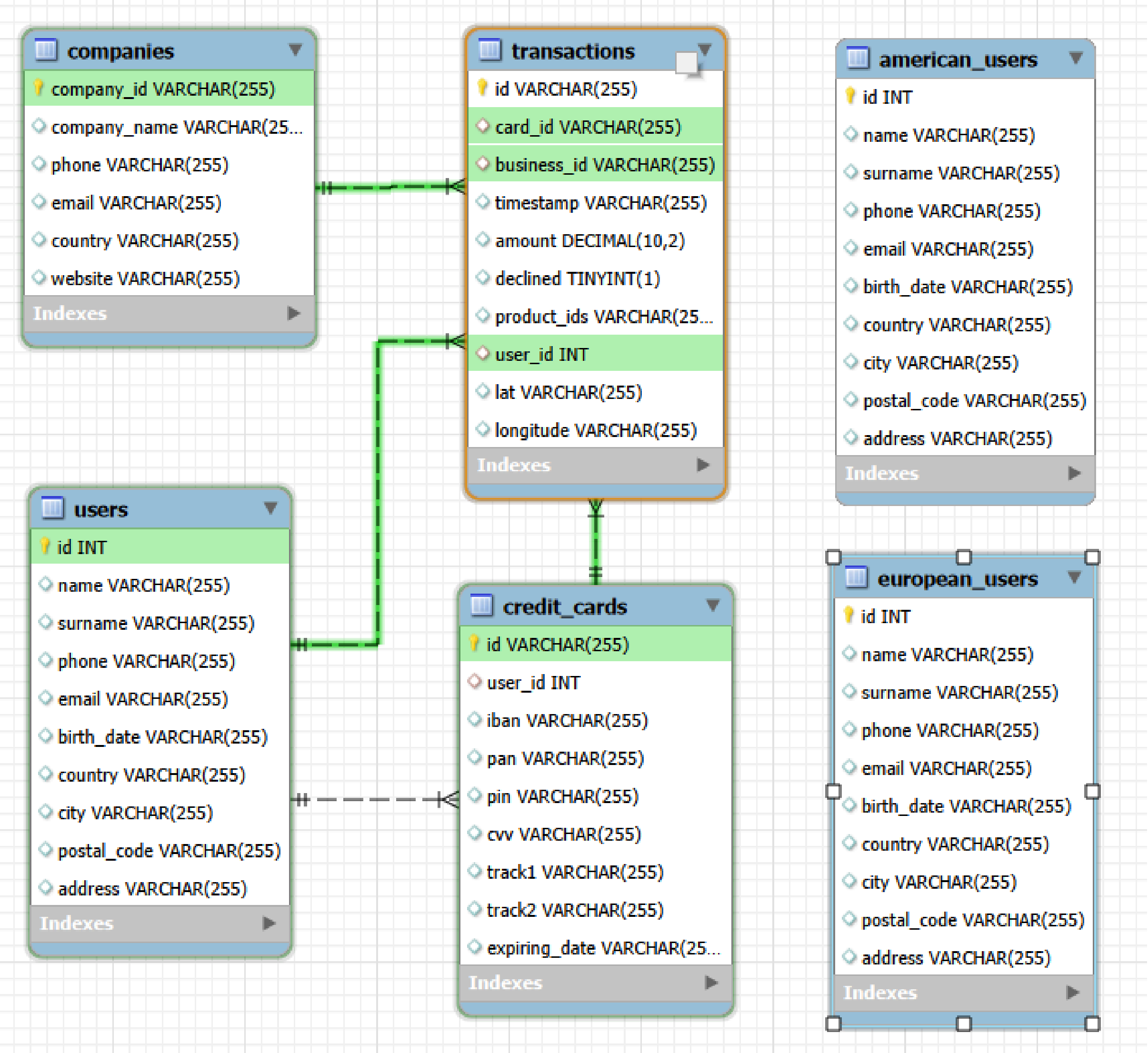


```sql
MODIFY COLUMN declined TINYINT(1),
MODIFY COLUMN user_id INT;

-- Añadimos foreign key con card_id
ALTER TABLE transactions
ADD CONSTRAINT fk_credit_cards
FOREIGN KEY (card_id)
REFERENCES credit_cards(id);

-- Añadimos foreign key con companies
ALTER TABLE transactions
ADD CONSTRAINT fk_companies_transactions
FOREIGN KEY (business_id)
REFERENCES companies(company_id);

-- Añadimos foreign key con users
ALTER TABLE transactions
ADD CONSTRAINT fk_user_transactions
FOREIGN KEY (user_id)
REFERENCES users(id);

SELECT * FROM transactions;
```

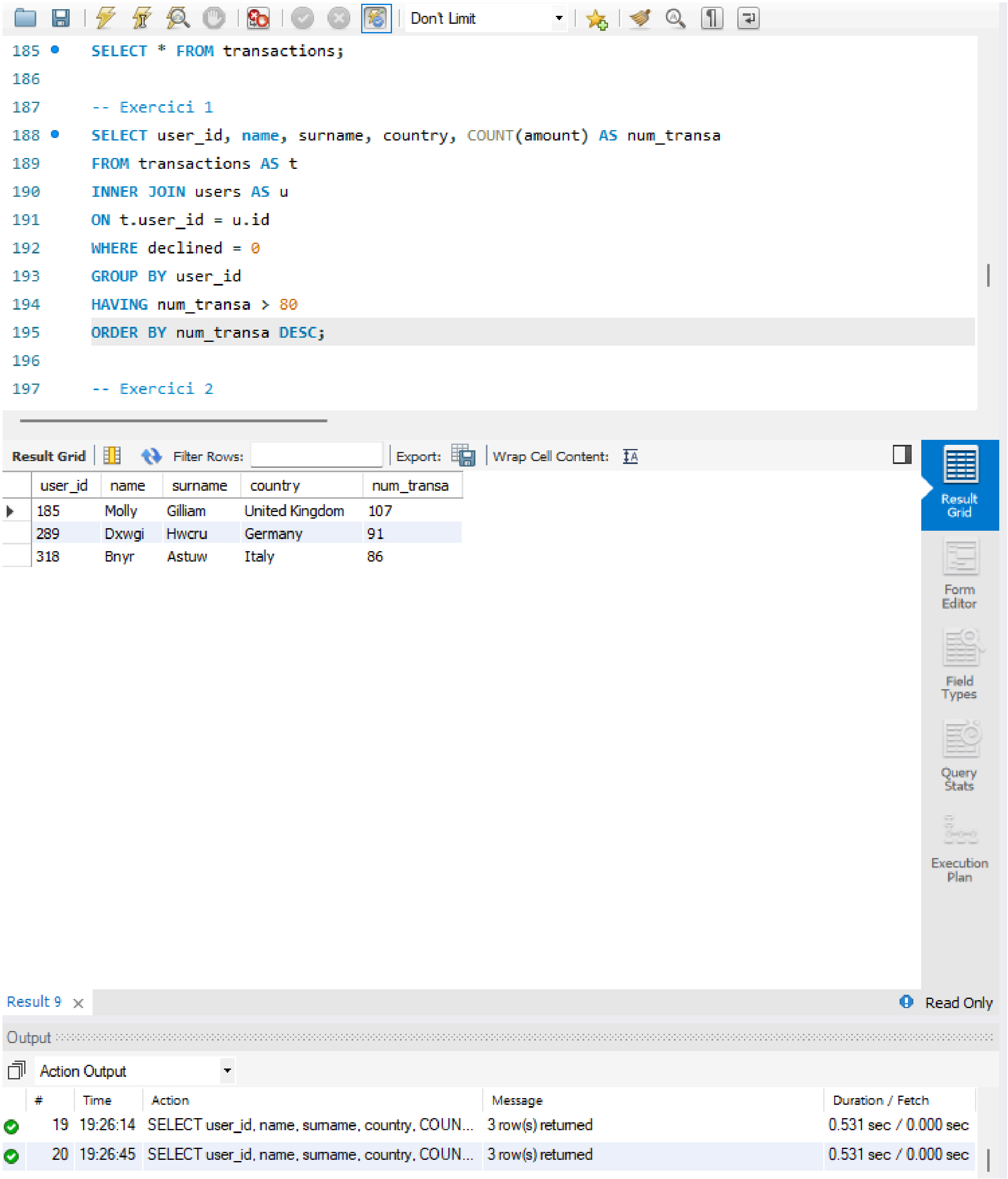
id	card_id	business_id	timestamp	amount	declined	product_ids
00043A49-2949-494B-A5DD-A5BAE3BB19DD	CcS-9294	b-2458	2024-08-28 07:16:46	395.43	0	16, 26, 97,
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	2016-12-21 20:07:18	155.63	0	66, 69, 87
00045D6B-ED2E-4F2F-8186-CEE074D875D0	CcS-6699	b-2390	2020-07-14 15:37:45	326.01	0	30, 11, 16,
000481C3-1C26-4FEF-83A0-4CD0EB004BBD	CcS-6696	b-2230	2017-09-04 19:44:53	161.60	0	72
00051AA4-9CBE-4268-B070-C38062A1B3E2	CcS-7606	b-2266	2017-01-05 18:19:25	148.91	0	18
0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	CcU-3358	b-2598	2023-09-23 04:51:43	294.59	0	35, 33, 19
0009A151-9BCF-4E31-9053-A468FF77FAAB	CcS-7509	b-2546	2023-12-31 00:06:36	383.63	0	93, 55, 28,
0009D494-6245-4DF9-955D-2C084191CFFB	CcS-8483	b-2526	2017-07-18 07:52:02	197.80	0	55, 8, 72
000A1DEC-CDB6-4AB2-A619-71DAB8D4A262	CcS-6467	b-2558	2018-09-08 05:29:58	339.94	0	46, 56, 73

	#	Time	Action	Message	Duration / Fetch
✓	16	19:21:56	SELECT * FROM companies	100 row(s) returned	0.000 sec / 0.000 sec
✓	17	19:23:43	SELECT * FROM transactions	100000 row(s) returned	0.015 sec / 0.219 sec

En la siguiente figura se muestran las relaciones entre las 6 tablas. Podemos observar una relación n-to-1 entre las tablas transactions y companies, unidas por transactions.business_id=companies_id. Esto se explica a que una misma compañía puede haber realizado diversas transacciones. También observamos la relación n-to-1 entre las tablas transactions y credit_cards, unidas por transactions.card_id=credit_cards.id; las tablas transactions y users, unidas por transactions.user_id=users.id; y entre las tablas credit_cards y users, unidas por credit_cards.user_id=users.id . Esto último indica que un usuario puede tener múltiples tarjetas de crédito.


companies
company_id VARCHAR(255)
company_name VARCHAR(25...
phone VARCHAR(255)
email VARCHAR(255)
country VARCHAR(255)
website VARCHAR(255)
Indexes
transactions
id VARCHAR(255)
card_id VARCHAR(255)
business_id VARCHAR(255)
timestamp VARCHAR(255)
amount DECIMAL(10,2)
declined TINYINT(1)
product_ids VARCHAR(25...
user_id INT
lat VARCHAR(255)
longitude VARCHAR(255)
Indexes
american_users
id INT
name VARCHAR(255)
surname VARCHAR(255)
phone VARCHAR(255)
email VARCHAR(255)
birth_date VARCHAR(255)
country VARCHAR(255)
city VARCHAR(255)
postal_code VARCHAR(255)
address VARCHAR(255)
Indexes
users
id INT
name VARCHAR(255)
surname VARCHAR(255)
phone VARCHAR(255)
email VARCHAR(255)
birth_date VARCHAR(255)
country VARCHAR(255)
city VARCHAR(255)
postal_code VARCHAR(255)
address VARCHAR(255)
Indexes
credit_cards
id VARCHAR(255)
user_id INT
iban VARCHAR(255)
pan VARCHAR(255)
pin VARCHAR(255)
cvv VARCHAR(255)
track1 VARCHAR(255)
track2 VARCHAR(255)
expiring_date VARCHAR(25...
Indexes
european_users
id INT
name VARCHAR(255)
surname VARCHAR(255)
phone VARCHAR(255)
email VARCHAR(255)
birth_date VARCHAR(255)
country VARCHAR(255)
city VARCHAR(255)
postal_code VARCHAR(255)
address VARCHAR(255)
Indexes

Una vez el esquema de estrella ha sido elaborado, realizamos la primera consulta de todos los usuarios con más de 80 transacciones.



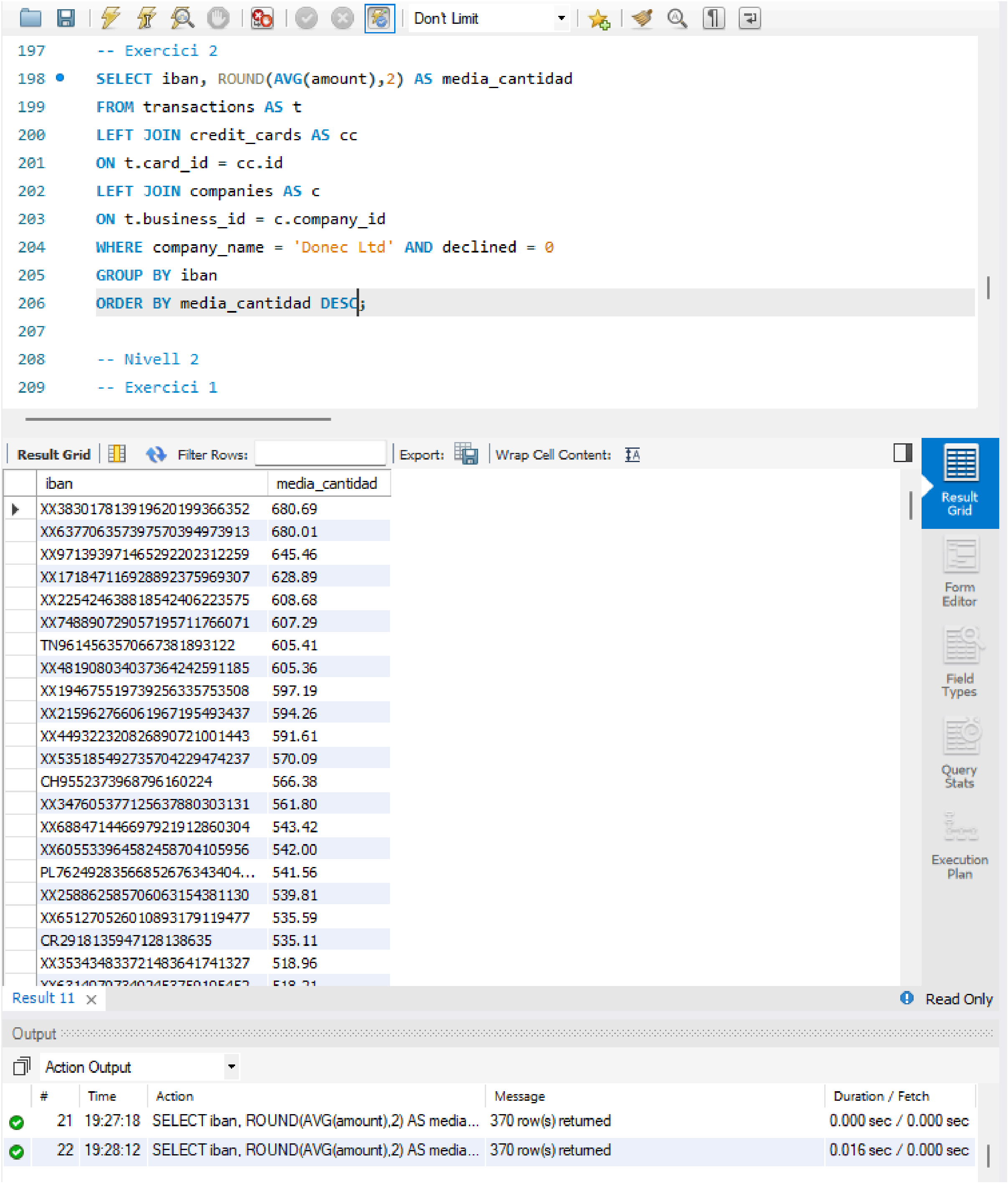
```sql
SELECT * FROM transactions;

-- Exercici 1
SELECT user_id, name, surname, country, COUNT(amount) AS num_transa
FROM transactions AS t
INNER JOIN users AS u
ON t.user_id = u.id
WHERE declined = 0
GROUP BY user_id
HAVING num_transa > 80
ORDER BY num_transa DESC;

-- Exercici 2
```

user_id	name	surname	country	num_transa
185	Molly	Gilliam	United Kingdom	107
289	Dxwgi	Hwcru	Germany	91
318	Bnyr	Astuw	Italy	86

#	Time	Action	Message	Duration / Fetch
19	19:26:14	SELECT user_id, name, surname, country, COUN...	3 row(s) returned	0.531 sec / 0.000 sec
20	19:26:45	SELECT user_id, name, surname, country, COUN...	3 row(s) returned	0.531 sec / 0.000 sec

Exercici 2

Hacemos la segunda consulta para la media de cantidad de cada tarjeta para la compañía indicada.



```sql
-- Exercici 2
SELECT iban, ROUND(AVG(amount),2) AS media_cantidad
FROM transactions AS t
LEFT JOIN credit_cards AS cc
ON t.card_id = cc.id
LEFT JOIN companies AS c
ON t.business_id = c.company_id
WHERE company_name = 'Donec Ltd' AND declined = 0
GROUP BY iban
ORDER BY media_cantidad DESC;

-- Nivell 2
-- Exercici 1
```

iban	media_cantidad
XX383017813919620199366352	680.69
XX637706357397570394973913	680.01
XX971393971465292202312259	645.46
XX171847116928892375969307	628.89
XX225424638818542406223575	608.68
XX748890729057195711766071	607.29
TN9614563570667381893122	605.41
XX481908034037364242591185	605.36
XX194675519739256335753508	597.19
XX215962766061967195493437	594.26
XX449322320826890721001443	591.61
XX535185492735704229474237	570.09
CH9552373968796160224	566.38
XX347605377125637880303131	561.80
XX688471446697921912860304	543.42
XX605533964582458704105956	542.00
PL76249283566852676343404...	541.56
XX258862585706063154381130	539.81
XX651270526010893179119477	535.59
CR2918135947128138635	535.11
XX353434833721483641741327	518.96

Result 11

Read Only

Output

Action Output

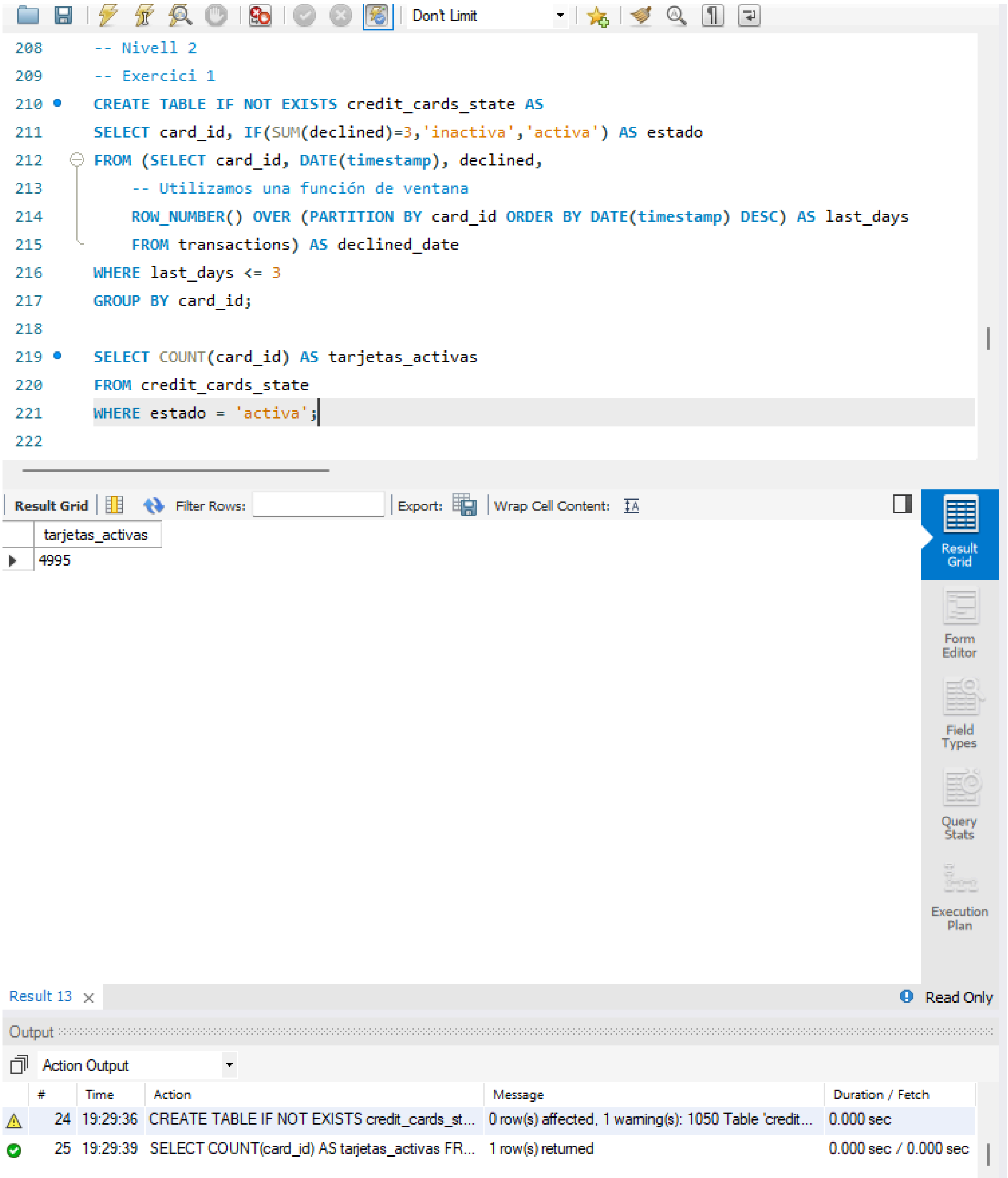
#	Time	Action	Message	Duration / Fetch
21	19:27:18	SELECT iban, ROUND(AVG(amount),2) AS media...	370 row(s) returned	0.000 sec / 0.000 sec
22	19:28:12	SELECT iban, ROUND(AVG(amount),2) AS media...	370 row(s) returned	0.016 sec / 0.000 sec

Nivell 2

Exercici 1

Para elaborar la tabla indicada, credit_card_state, utilizamos la función de tabla ROW_NUMBER() con PARTITION BY, lo que nos permite tener una numeración de las fechas de transacciones que reinicia para cada tarjeta. De esta tabla resultado, se decidió si la tarjeta estaba activa o no sumando declined en los tres últimos días (last_days <= 3), de tal manera que si había estado declinada 3 veces (SUM(declined)=3), la tarjeta se considera inactiva.

Con la nueva tabla credit_card_state, se realizó la consulta indicada.



```sql
-- Nivell 2
-- Exercici 1
CREATE TABLE IF NOT EXISTS credit_cards_state AS
SELECT card_id, IF(SUM(declined)=3,'inactiva','activa') AS estado
FROM (SELECT card_id, DATE(timestamp), declined,
    -- Utilizamos una función de ventana
    ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY DATE(timestamp) DESC) AS last_days
    FROM transactions) AS declined_date
WHERE last_days <= 3
GROUP BY card_id;

SELECT COUNT(card_id) AS tarjetas_activas
FROM credit_cards_state
WHERE estado = 'activa';
```

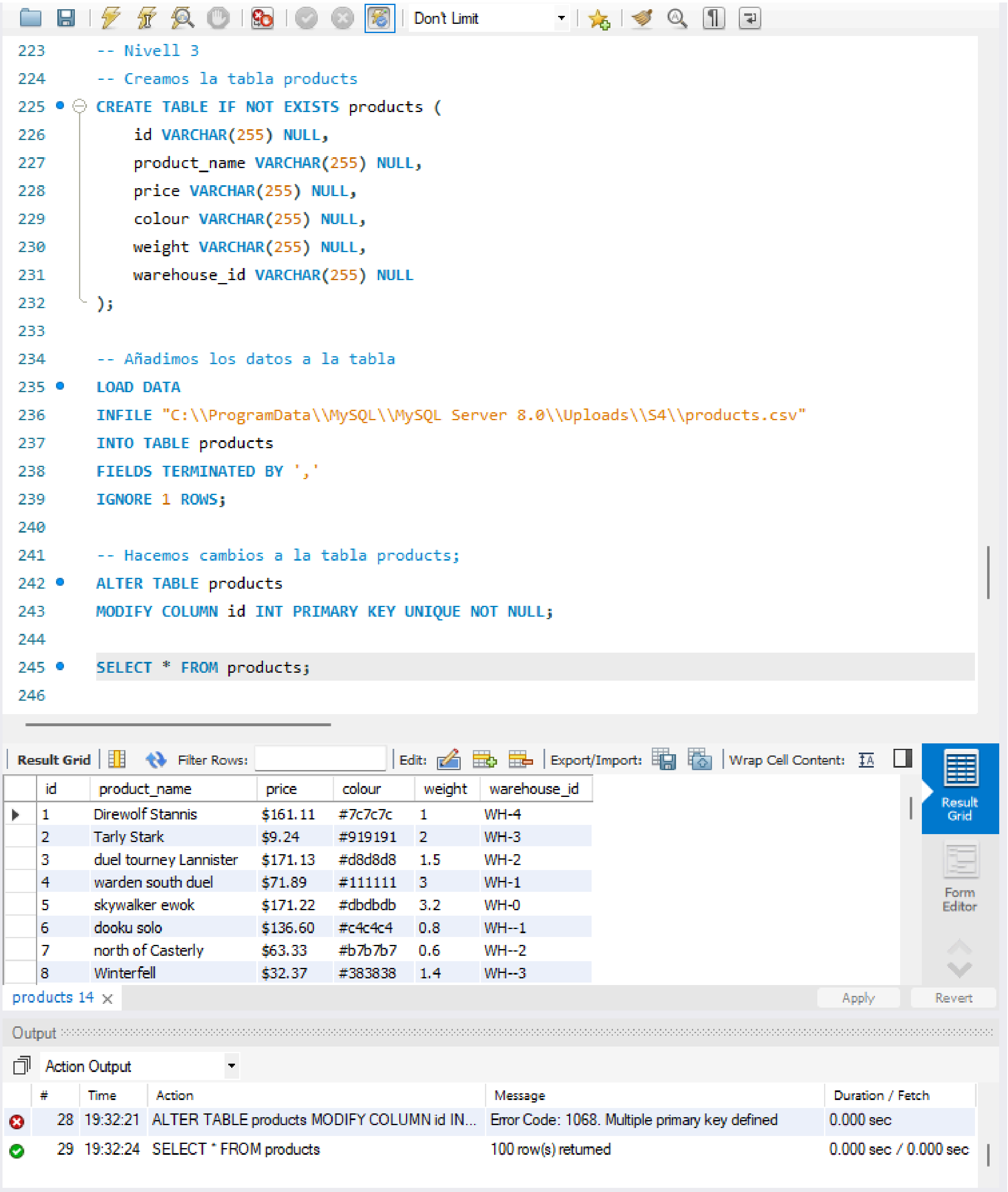
tarjetas_activas
4995

	#	Time	Action	Message	Duration / Fetch
⚠	24	19:29:36	CREATE TABLE IF NOT EXISTS credit_cards_st...	0 row(s) affected, 1 warning(s): 1050 Table 'credit...	0.000 sec
✓	25	19:29:39	SELECT COUNT(card_id) AS tarjetas_activas FR...	1 row(s) returned	0.000 sec / 0.000 sec

Nivell 3

Exercici 1

Antes de crear la nueva tabla, introducimos la tabla con datos de products.csv.

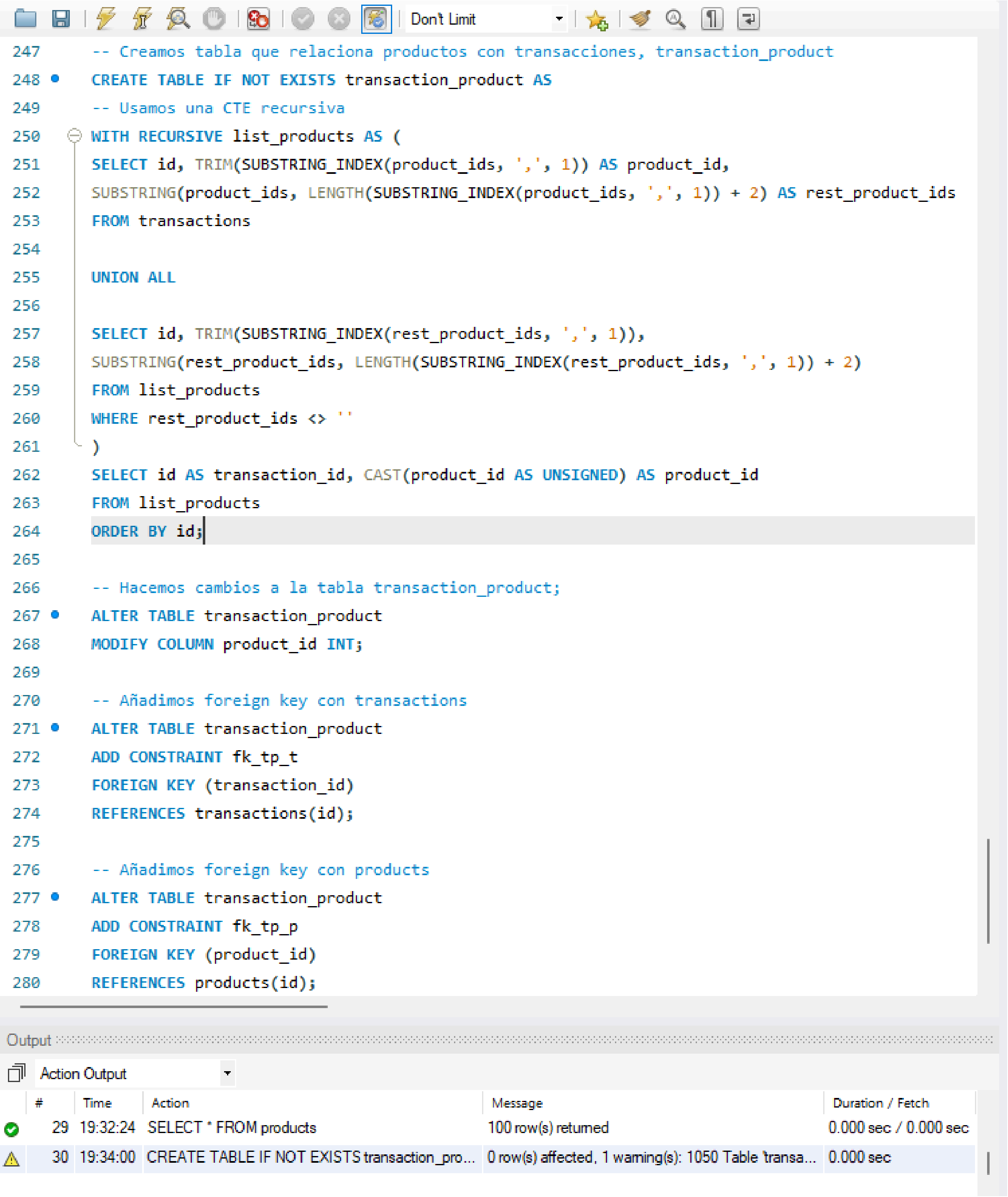


```sql
-- Nivell 3
-- Creamos la tabla products
CREATE TABLE IF NOT EXISTS products (
    id VARCHAR(255) NULL,
    product_name VARCHAR(255) NULL,
    price VARCHAR(255) NULL,
    colour VARCHAR(255) NULL,
    weight VARCHAR(255) NULL,
    warehouse_id VARCHAR(255) NULL
);

-- Añadimos los datos a la tabla
LOAD DATA
INFILE "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\S4\\products.csv"
INTO TABLE products
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;

-- Hacemos cambios a la tabla products;
ALTER TABLE products
MODIFY COLUMN id INT PRIMARY KEY UNIQUE NOT NULL;

SELECT * FROM products;
```

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

	#	Time	Action	Message	Duration / Fetch
✗	28	19:32:21	ALTER TABLE products MODIFY COLUMN id IN...	Error Code: 1068. Multiple primary key defined	0.000 sec
✓	29	19:32:24	SELECT * FROM products	100 row(s) returned	0.000 sec / 0.000 sec

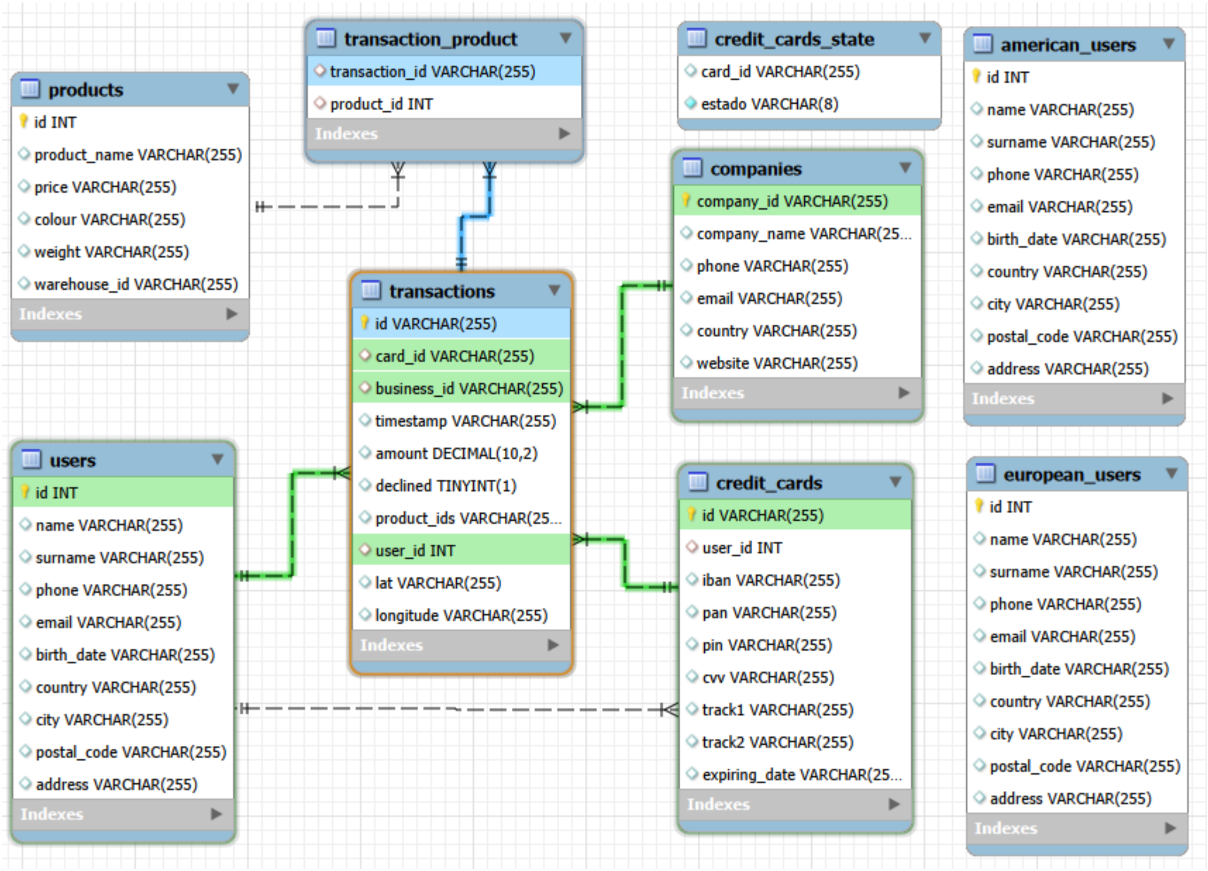
Ahora con ya todas las tablas, elaboramos la tabla transaction_product. Para elaborarla, usamos una CTE recursiva, con la cual extraemos el primer índice de la lista de productos de cada fila de transacción con TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) y guardabamos el resto con SUBSTRING(product_ids, LENGTH(SUBSTRING_INDEX(product_ids, ',', 1)) + 2). Continuamos extrayendo índices de productos hasta que la lista esté vacía.



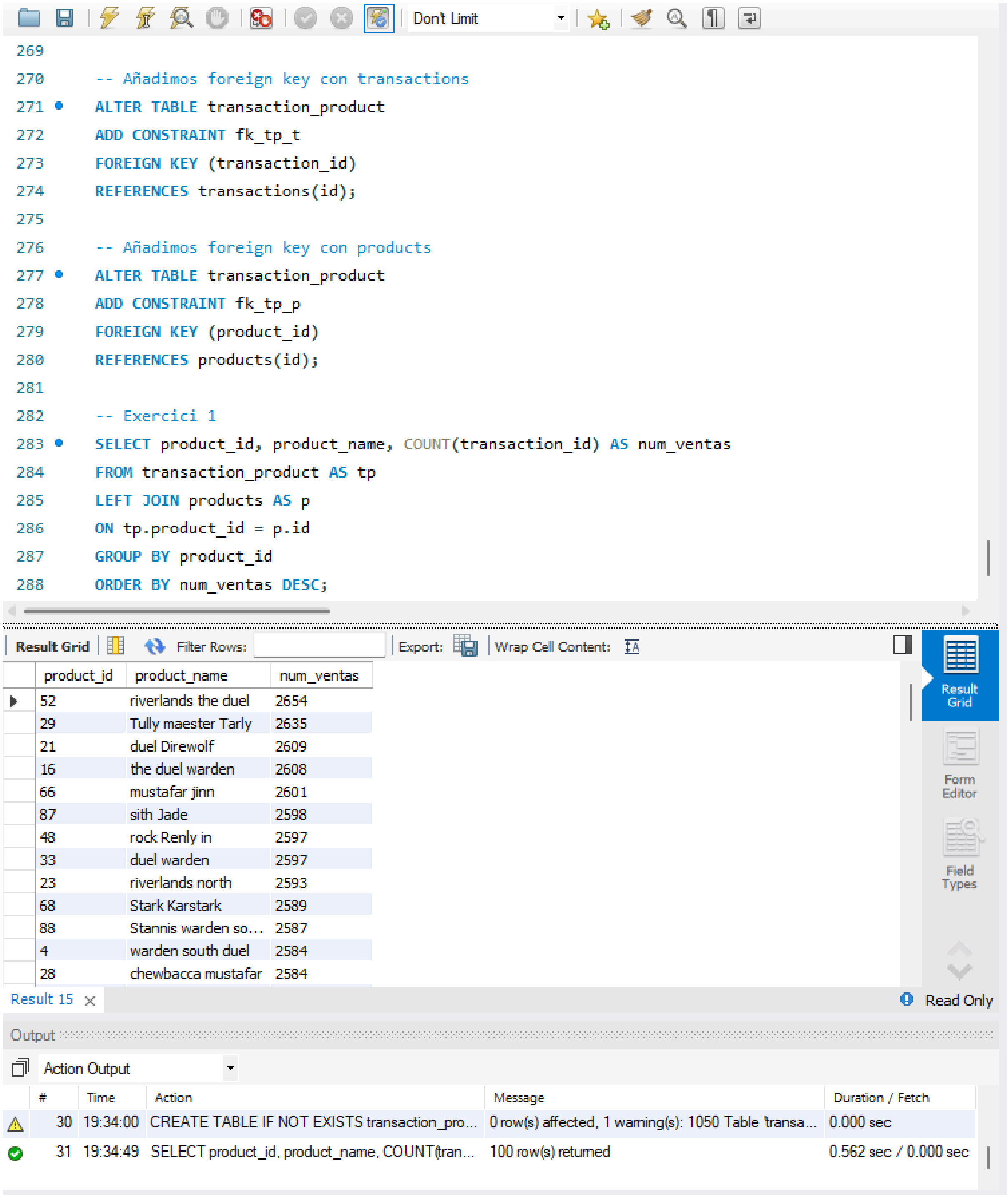
```sql
-- Creamos tabla que relaciona productos con transacciones, transaction_product
CREATE TABLE IF NOT EXISTS transaction_product AS
-- Usamos una CTE recursiva
WITH RECURSIVE list_products AS (
SELECT id, TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) AS product_id,
SUBSTRING(product_ids, LENGTH(SUBSTRING_INDEX(product_ids, ',', 1)) + 2) AS rest_product_ids
FROM transactions

UNION ALL

SELECT id, TRIM(SUBSTRING_INDEX(rest_product_ids, ',', 1)),
SUBSTRING(rest_product_ids, LENGTH(SUBSTRING_INDEX(rest_product_ids, ',', 1)) + 2)
FROM list_products
WHERE rest_product_ids <> ''
)
SELECT id AS transaction_id, CAST(product_id AS UNSIGNED) AS product_id
FROM list_products
ORDER BY id;

-- Hacemos cambios a la tabla transaction_product;
ALTER TABLE transaction_product
MODIFY COLUMN product_id INT;

-- Añadimos foreign key con transactions
ALTER TABLE transaction_product
ADD CONSTRAINT fk_tp_t
FOREIGN KEY (transaction_id)
REFERENCES transactions(id);

-- Añadimos foreign key con products
ALTER TABLE transaction_product
ADD CONSTRAINT fk_tp_p
FOREIGN KEY (product_id)
REFERENCES products(id);
```

	#	Time	Action	Message	Duration / Fetch
✓	29	19:32:24	SELECT * FROM products	100 row(s) returned	0.000 sec / 0.000 sec
⚠	30	19:34:00	CREATE TABLE IF NOT EXISTS transaction_pro...	0 row(s) affected, 1 warning(s): 1050 Table 'transa...	0.000 sec

Tras los cambios indicados, obtuvimos el siguiente diagrama de estrella.



Elaboramos la consulta para saber el número de veces que se ha vendido cada producto.



```sql

-- Añadimos foreign key con transactions
ALTER TABLE transaction_product
ADD CONSTRAINT fk_tp_t
FOREIGN KEY (transaction_id)
REFERENCES transactions(id);

-- Añadimos foreign key con products
ALTER TABLE transaction_product
ADD CONSTRAINT fk_tp_p
FOREIGN KEY (product_id)
REFERENCES products(id);

-- Exercici 1
SELECT product_id, product_name, COUNT(transaction_id) AS num_ventas
FROM transaction_product AS tp
LEFT JOIN products AS p
ON tp.product_id = p.id
GROUP BY product_id
ORDER BY num_ventas DESC;
```

product_id	product_name	num_ventas
52	riverlands the duel	2654
29	Tully maester Tarly	2635
21	duel Direwolf	2609
16	the duel warden	2608
66	mustafar jinn	2601
87	sith Jade	2598
48	rock Renly in	2597
33	duel warden	2597
23	riverlands north	2593
68	Stark Karstark	2589
88	Stannis warden so...	2587
4	warden south duel	2584
28	chewbacca mustafar	2584

	#	Time	Action	Message	Duration / Fetch
⚠	30	19:34:00	CREATE TABLE IF NOT EXISTS transaction_pro...	0 row(s) affected, 1 warning(s): 1050 Table 'transa...	0.000 sec
✓	31	19:34:49	SELECT product_id, product_name, COUNT(tran...	100 row(s) returned	0.562 sec / 0.000 sec

Revisión peer-to-peer

Revisado por Minu Campoy

- Tener en cuenta que al unir american_users y european_users en la tabla users, se pierde información si no se crea una nueva columna que indique de qué tabla procede cada usuario. No es un problema en esta actividad porque ningún ejercicio pide tener en cuenta la diferencia, pero se debe considerar por si fuera necesario.
- Se podría conectar la tabla users con american_users y european_users, como forma de mantener la separación que se pierde en users haciendo un JOIN con cualquiera de las dos.
- También se podría unir la tabla credit_cards con credit_cards_state.