Tasca S4.01. Creació de Base de Dades

Fecha 04/11/2025
Andrey Ismagilov

Nivel 1

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules.

Vamos a crear la base de datos **transactions_sp4.**

Hay 6 archivos en el formato CSV:

1. **transactions.csv** contiene la información de cada venta realizada en línea. Usando este archivo podemos crear la tabla **transaction**.

```
id;card_id;business_id;timestamp;amount;declined;product_ids;user_id;lat;longitude
CDDA7E40-544D-47BB-A4ED-671DD8A950D9;CcS-6894;b-2466;2018-12-12 08:05:17;161.88;0;75, 73, 98;2313;59.62050974356148;16.559977155728436
09456357-8E9B-475A-8257-87A023699964;CcS-5135;b-2342;2024-05-20 22:49:08;171.13;0;3;554;45.76458841901318;4.843056518287656
C47C7C84-C174-4973-A76B-825A9DB5582A;CcS-8415;b-2250;2018-11-04 23:16:18;497.29;0;92, 85, 36, 23;3834;52.06849608409912;4.301099382555438
2FB526AA-3844-4DDF-AC09-2EBCC255EE76;CcS-6553;b-2610;2022-06-17 09:17:25;344.15;0;5, 27;1972;39.47598513179192;-0.3764194439184784
0D877407-B85E-40D2-9229-6BD6F2B8DF0A;CcS-7678;b-2454;2020-11-29 07:29:16;435.1;0;38, 32, 82, 62;3097;51.43656985032189;5.478525648707596
```

Vemos que los valores se separan por punto y coma.
El encabezado del archivo tiene los nombres de la siguientes columnas:

1. **id** – el identificador de la transacción que será la variable **id** del tipo *VARCHAR(255)* en la tabla **transaction**. Será el **Primary Key** de la tabla **transaction.**
2. **card_id** – el identificador la tarjeta de crédito del tipo *VARCHAR(20)*. Será la **credit_card_id** en la tabla y tambien el **FOREIGN KEY** de la tabla **transaction** que estará vinculada con la variable **id** de la tabla **credit_cards** a continuación.
3. **business_id** – el identificador de la empresa que venda los productos en línea. Será el variable **company_id** del tipo *VARCHAR(20)* y el **FOREIGN KEY** de la tabla **transaction** que estará vinculada con la variable **id** de la tabla **companies** luego.
4. **timestamp –** timestamp de la transacción. Será la **timestamp** del tipo *VARCHAR(255).*
5. **amount –** importe de la transacción. La **amount** del tipo *DECIMAL(10, 2)*.
6. **declined** – es igual a **0** si la transacción realizada e igual a **1** si rechazada. Será la **declined** del tipo *TINYINT.*
7. **product_ids –** la lista de los productos incluidos en la transacción (venta). Se separan por coma. Será la **product_ids** del tipo *VARCHAR(255).*
8. **user_id –** el identificador del comprador. Será la **user_id** del tipo *INT* y el **FOREIGN KEY** de la tabla **transaction** que estará vinculada con la variable **id** de la tabla **data_users** luego.
9. **lat -** latitud del lugar donde se realizó la venta. Será la **lat** del tipo *FLOAT*.
10. **longitude -** longitud del lugar donde se realizó la venta. Será la **longitude** del tipo *FLOAT*.

2. **credit_cards.csv** guarda detalles sobre las tarjetas de crédito. Con este archivo creamos la tabla **credit_cards.**

```
id,user_id,iban,pan,pin,cvv,track1,track2,expiring_date
CcU-2938,275,TR301950312213576817638661,5424465566813633,3257,984,%B8383712448554646^WovsxejDpwiev^86041142?7,%B7653863056044187=8007163336?3,10/30/22
CcU-2945,274,DO28547637485374752165686 89,5142423821948828,9080,887,%B4621311609958661^UftuyfsSeimxn^0610628241?7,%B4149568437843501=5107140330?1,08/24/23
CcU-2952,273,BG45IVQL52710525608255,4556 453 55 5287,4598,438,%B2183285104307501^CddyytcUxwfdq^5907955430?9,%B6778580257827162=69068597400?7,06/29/21
CcU-2959,272,CR7242477244335841535,372461377349375,3583,667,%B7281111956795320^XocddijBckecd^09016253?3,%B4246154489281853=2805223916?8,02/24/23
CcU-2966,271,BG72LKTQ15627628377363,448566 886747 7265,4900,130,%B4728932322756223^JhlgvsuFbmwgj^72022894943?7,%B2318571115599881=8908215784?5,10/29/24
```

Vemos que los valores se separan por coma.
El encabezado del archivo tiene los nombres de la siguientes columnas:

1. **id –** el identificador de la tarjeta crédito que será la variable **id** del tipo *VARCHAR(20)* en la tabla **credit_cards** y el **Primary Key.**
2. **user_id -** el identificador del comprador. Será la **user_id** del tipo *INT*.
3. **iban -** un numero de cuenta bancaria vinculada a la tarjeta crédito, será la **iban** del tipo *VARCHAR(50)*.
4. **pan -** un número de la tarjeta que se encuentra impreso en el anverso (y a veces reverso) y que la identifica de forma única, será la **pan** del tipo *VARCHAR(50)*.
5. **pin -** un número de identificación personal de 4 dígitos que funciona como una clave secreta para autorizar transacciones, será la **pin** del tipo *VARCHAR(4)*.
6. **cvv -** un código de seguridad de 3 o 4 dígitos, será la **cvv** del tipo *VARCHAR(4)*.
7. **track1 –** los datos en la banda magnética de la tarjeta de crédito, será la **track1** del tipo *VARCHAR(255)*.
8. **track2 -** los datos en la banda magnética de la tarjeta de crédito, será la **track2** del tipo *VARCHAR(255)*.
9. **expiring_date -** la fecha de vencimiento de una tarjeta, será la **expiring_date** del tipo *VARCHAR(20)*.

3. **companies.csv** - las empresas que vendan sus productos en línea.

```
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
b-2234,Convallis In Incorporated,06 66 57 29 50,mauris.ut@aol.couk,Germany,https://cnn.com/user/110
b-2238,Ante Iaculis Nec Foundation,08 23 04 99 53,sed.dictum.proin@outlook.ca,New Zealand,https://netflix.com/settings
```

Vemos que los valores se separan por coma.
El encabezado del archivo tiene los nombres de la siguientes columnas:

1. **id** – el identificador de la empresa. Será el **Primary Key** de la tabla **companies** del tipo *VARCHAR(20)* .
2. **company_name** – nombre de la empresa. Será la **company_name** del tipo *VARCHAR(255)*.
3. **phone** – el numero de telefono. Será la **phone** del tipo *VARCHAR(15)*.
4. **email** – el correo electronico de la empresa. Será la **email** del tipo *VARCHAR(100)*.
5. **country** – el pais donde esta la empresa. Será la **country** del tipo *VARCHAR(100)*.
6. **website** – la pagina web de la empresa. Será la **website** del tipo *VARCHAR(255)*.

4. **european_users.csv** - la información detallada sobre compradores que viven en la zona europea.

```
id,name,surname,phone,email,birth_date,country,city,postal_code,address
151,Meghan,Hayden,0800 746 6747,arcu.vel@hotmail.ca,"Jul 2, 1980",United Kingdom,London,EC1A 1BB,Ap #432-4493 Aliquet Rd.
152,Hakeem,Alford,(0111) 367 0184,adipiscing.ligula@google.edu,"Sep 30, 1979",United Kingdom,Birmingham,B1 1AA,551-8930 Lobortis Street
153,Keegan,Pugh,(016977) 3851,sodales.nisi@aol.org,"Jul 27, 1994",United Kingdom,London,EC1A 1BB,Ap #312-5898 Consectetuer St.
154,Cooper,Bullock,(021) 2521 6627,et@outlook.net,"Nov 2, 1986",United Kingdom,Manchester,M1 1AE,872-1866 Pede Rd.
155,Joshua,Russell,055 4409 5286,justo.nec.ante@outlook.edu,"Jan 23, 1984",United Kingdom,Manchester,M1 1AE,Ap #285-4727 Auctor. Av.
```

Los valores se separan por coma.
El encabezado del archivo tiene los nombres de la siguientes columnas:

1. **id** – el identificador del comprador. Será el **Primary Key** de la tabla **data_user** del tipo *INT* .
2. **name** – el nombre del comprador. Será la **name** del tipo *VARCHAR(100)*.
3. **surname** – el apellido del comprador. Será la **surname** del tipo *VARCHAR(100)*.
4. **phone** – el numero de telefono del comprador. Será la **phone** del tipo *VARCHAR(150)*.
5. **email** – el correo electronico del comprador. Será la **email** del tipo *VARCHAR(150)*.
6. **birth_date –** el cumpleaños del comprador. Será la **birth_date** del tipo *VARCHAR(100)*.
7. **country** – el pais donde esta la empresa. Será la **country** del tipo *VARCHAR(150)*.
8. **city –** la ciudad donde vive el comprador. Será la **city** del tipo *VARCHAR(150)*.
9. **postal_code – el codico postal del comprador.** Será la **postal_code** del tipo *VARCHAR(100)*.
10. **address** – la dirección del comprador. Será la **address** del tipo *VARCHAR(255)*.

5. **american_users.csv** - la información detallada sobre compradores que viven en la zona americana.

```
id,name,surname,phone,email,birth_date,country,city,postal_code,address
1,Zeus,Gamble,1-282-581-0551,interdum.enim@protonmail.edu,"Nov 17, 1985",United States,New York,10001,348-7818 Sagittis St.
2,Garrett,Mcconnell,(718) 257-2412,integer.vitae.nibh@protonmail.org,"Aug 23, 1992",United States,Philadelphia,19101,903 Sit Ave
3,Ciaran,Harrison,(522) 598-1365,interdum.feugiat@aol.org,"Apr 29, 1998",United States,Houston,77001,736-2063 Tellus St.
4,Howard,Stafford,1-411-740-3269,ornare.egestas@icloud.edu,"Feb 18, 1989",United States,Phoenix,85001,Ap #545-2244 Erat. Rd.
5,Hayfa,Pierce,1-554-541-2077,et.malesuada.fames@hotmail.org,"Sep 26, 1998",United States,Philadelphia,19101,341-2821 Ultrices Av.
```

El encabezado del archivo tiene los nombres iguales que el archivo **european_users.csv** por eso combinaremos la tabla **american_users** y la **european_users** a la unica tabla **data_users** luego**.**

6. **products.csv** -almacena la información sobre cada producto que se venda en línea.

```
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
```

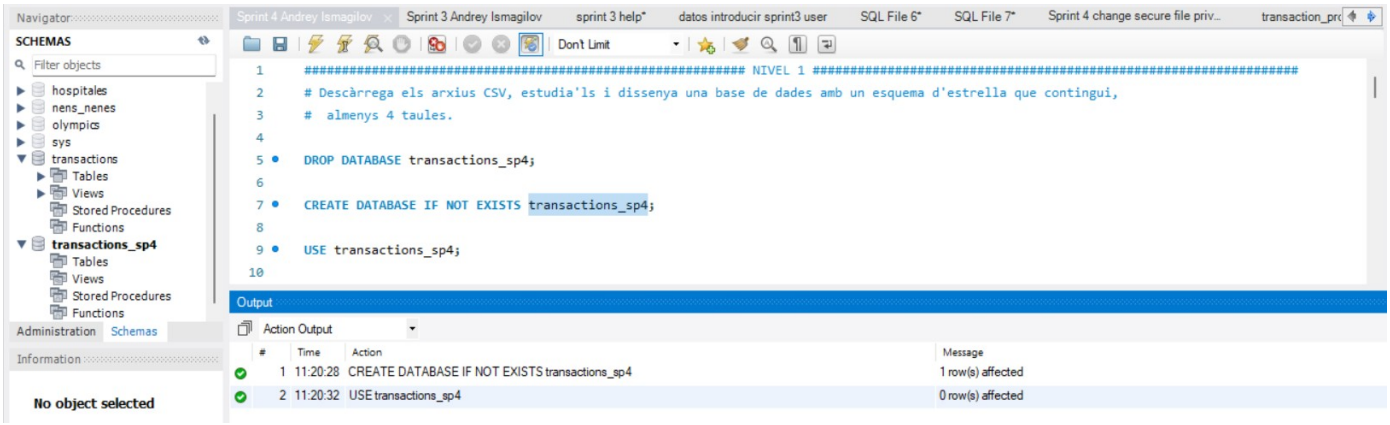
Vemos que los valores se separan por coma.
El encabezado del archivo tiene los nombres de la siguientes columnas:

1. **id** – el codico del producto. Será la variable **id** de la tabla **products** del tipo *INT* y el **PRIMARY KEY** de la tabla.
2. **product_name –** el nombre del producto. Será la variable **product_name** del tipo *VARCHAR(50)*.
3. **price –** el precio del producto. Será la variable **price** *del tipo DECIMAL(10, 2)***.**
4. **colour –** el codico de color del producto. Será la variable **colour** del tipo *VARCHAR(20)*.
5. **weight –** el peso del producto. Será la variable **weight** del tipo *VARCHAR(20).*
6. **warehouse_id –** el identificador de almacén. Será la variable **warehouse_id** del tipo *VARCHAR(20)*.

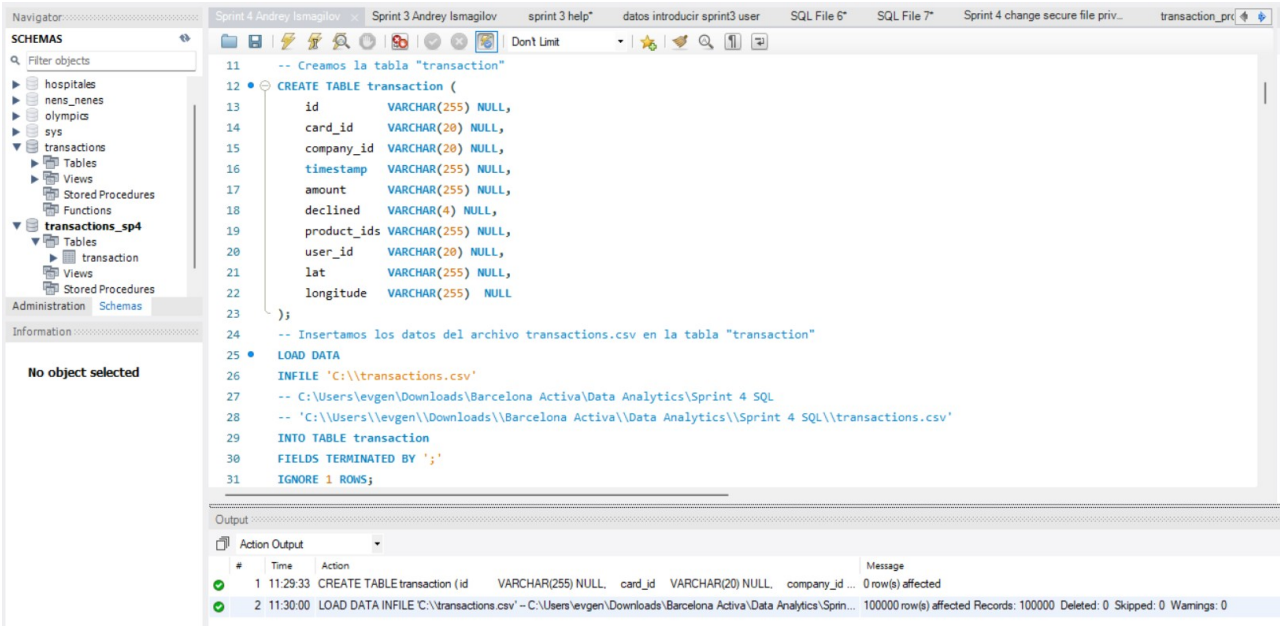
Eligimos 5 de ellos: el **transactions.csv** que será la tabla **transaction** (la tabla de hechos de la esquema de estrella), el **credit_cards.csv** (será la tabla **credit_cards**), el **companies.csv** (la tabla **companies**), el **european_users.csv** y el **american_users.csv** que formaran la tabla **data_user**.

Al principio los tipos de columas de todas las tablas seran de *VARCHAR* para que los datos vaya cargar sin errores. Luego cambiamos los tipos de columnas con datos numericos.

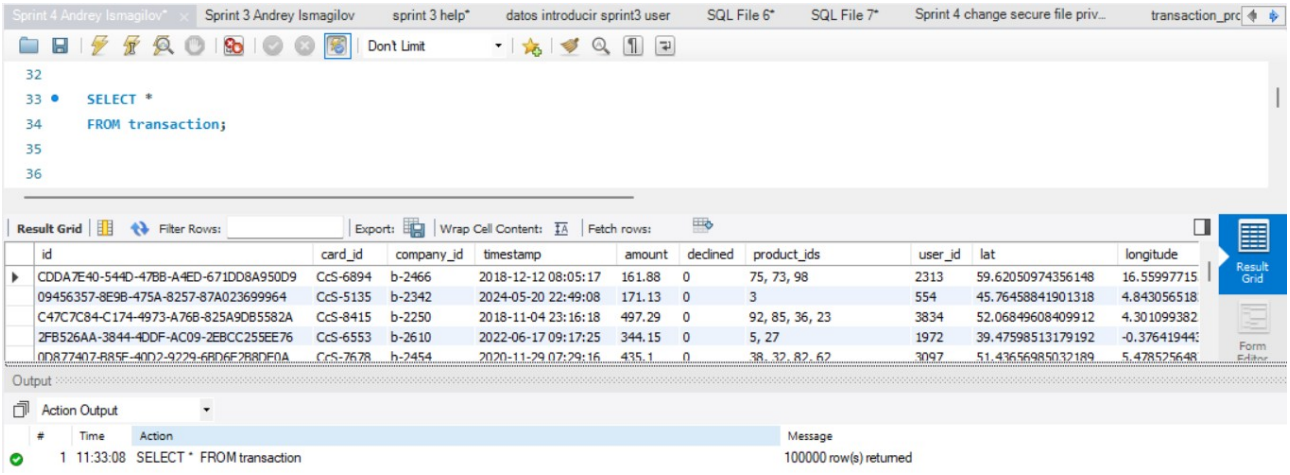
Paso 1. Creamos la database **transactions_sp4.**



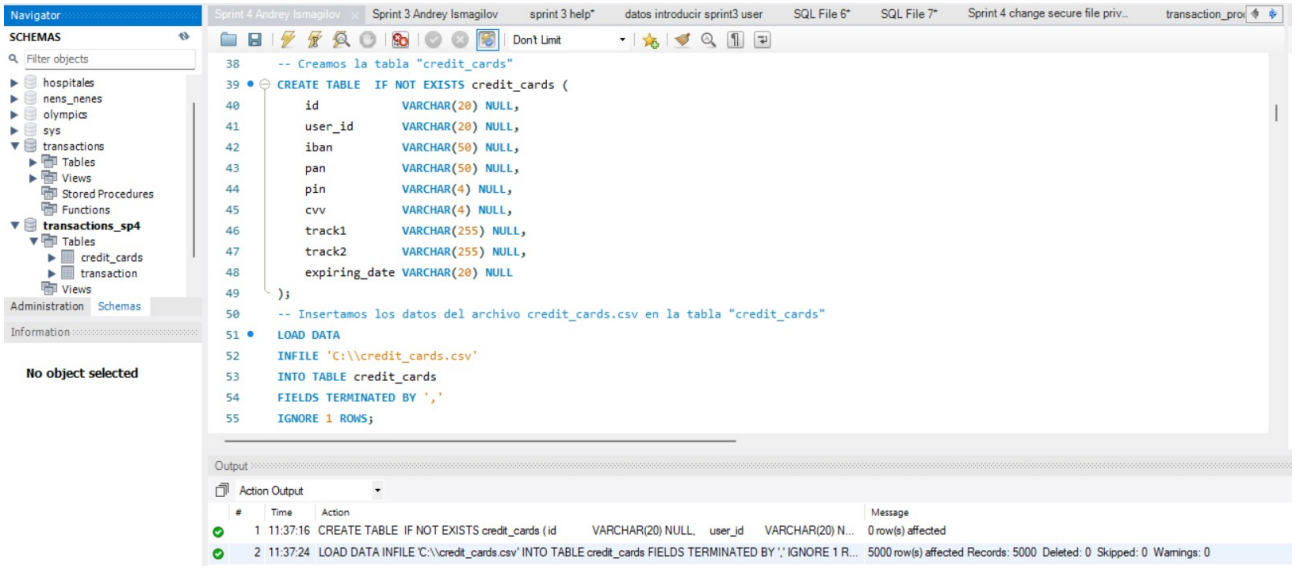
Paso 2. Creamos la tabla **transaction** y luego insertamos los datos del archivo **transactions.csv** en la tabla.



Se ve, que la **transaction** tiene 100000 registros.

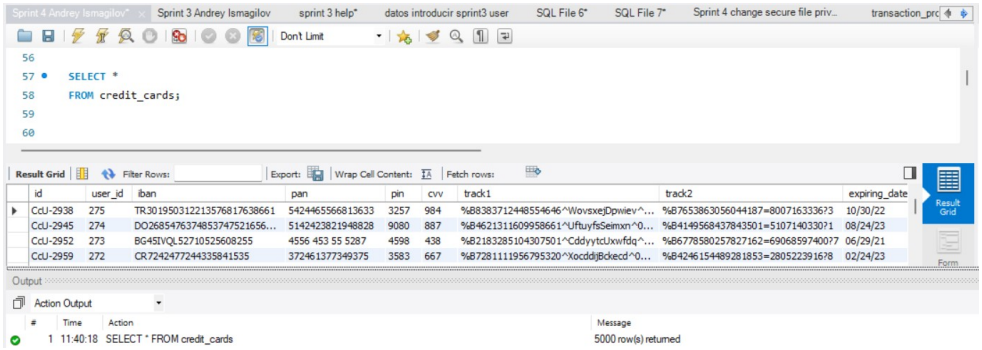


Paso 3. Creamos la tabla **credit_cards** e insertamos los datos del archivo **credit_cards.csv**.



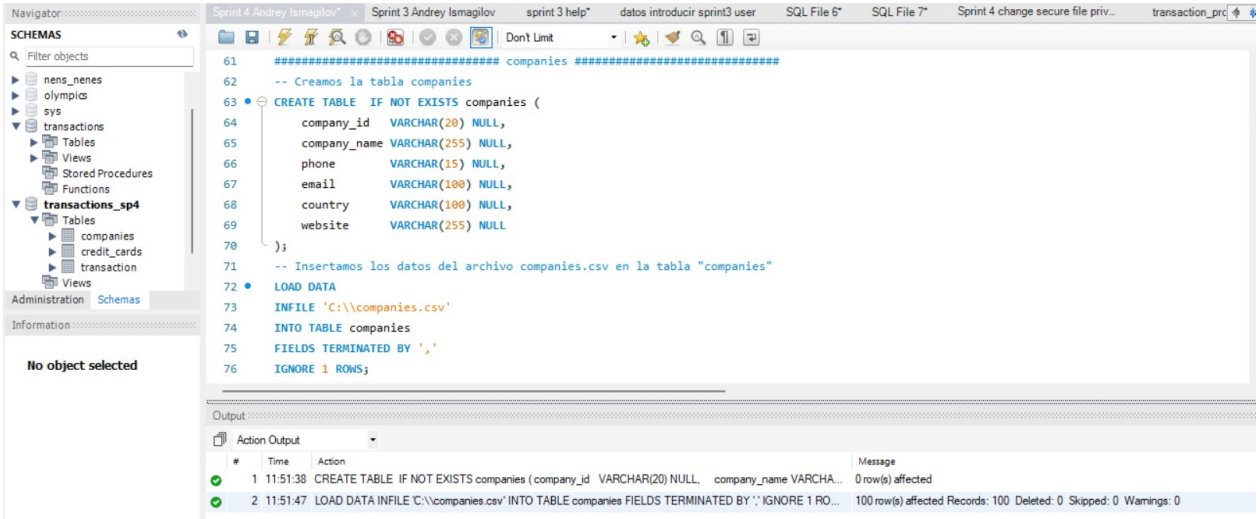
```
-- Creamos la tabla "credit_cards"
CREATE TABLE  IF NOT EXISTS credit_cards (
    id            VARCHAR(20) NULL,
    user_id       VARCHAR(20) NULL,
    iban          VARCHAR(50) NULL,
    pan           VARCHAR(50) NULL,
    pin           VARCHAR(4) NULL,
    cvv           VARCHAR(4) NULL,
    track1        VARCHAR(255) NULL,
    track2        VARCHAR(255) NULL,
    expiring_date VARCHAR(20) NULL
);
-- Insertamos los datos del archivo credit_cards.csv en la tabla "credit_cards"
LOAD DATA
INFILE 'C:\\credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;
```

La tabla **credit_cards** tiene 5000 filas.



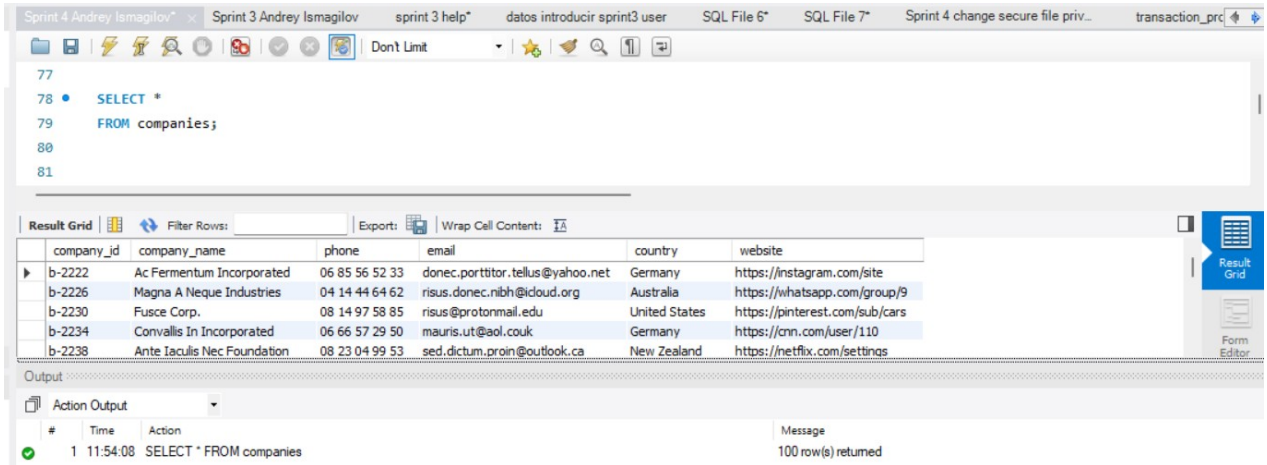
```
SELECT *
FROM credit_cards;
```

Paso 4. Creamos la tabla **companies** e insertamos los datos del archivo **companies.csv**.



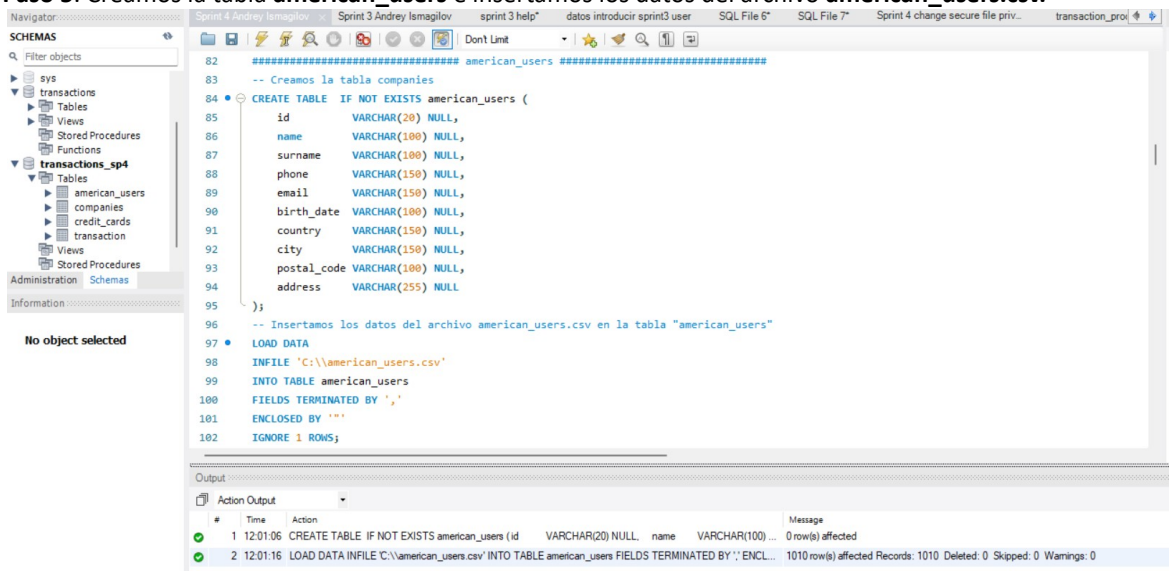
```
################################ companies ############################
-- Creamos la tabla companies
CREATE TABLE  IF NOT EXISTS companies (
    company_id   VARCHAR(20) NULL,
    company_name VARCHAR(255) NULL,
    phone        VARCHAR(15) NULL,
    email        VARCHAR(100) NULL,
    country      VARCHAR(100) NULL,
    website      VARCHAR(255) NULL
);
-- Insertamos los datos del archivo companies.csv en la tabla "companies"
LOAD DATA
INFILE 'C:\\companies.csv'
INTO TABLE companies
FIELDS TERMINATED BY ','
IGNORE 1 ROWS;
```

La tabla **companies** tiene **100** filas.



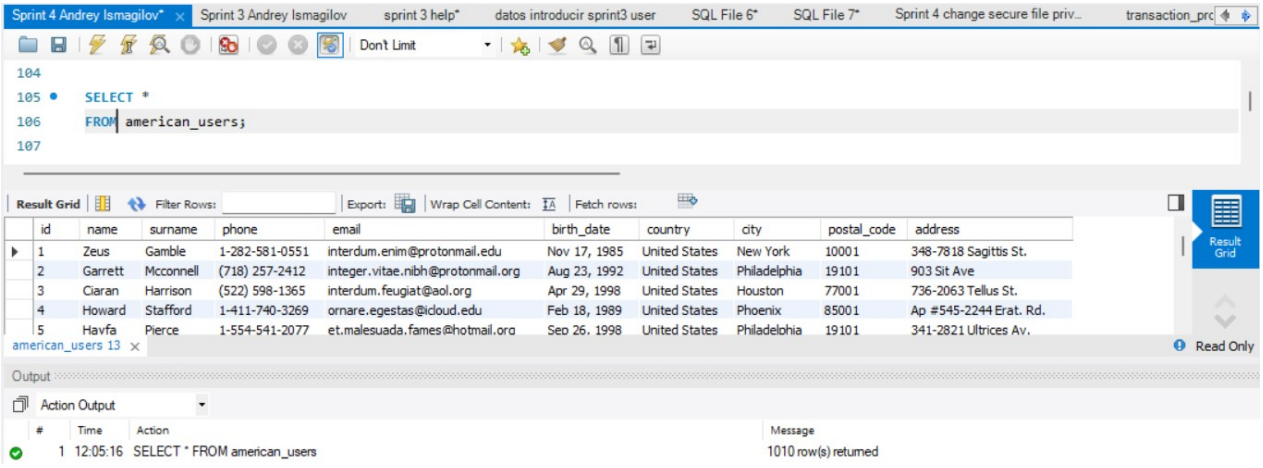
```
SELECT *
FROM companies;
```

Paso 5. Creamos la tabla **american_users** e insertamos los datos del archivo **american_users.csv.**



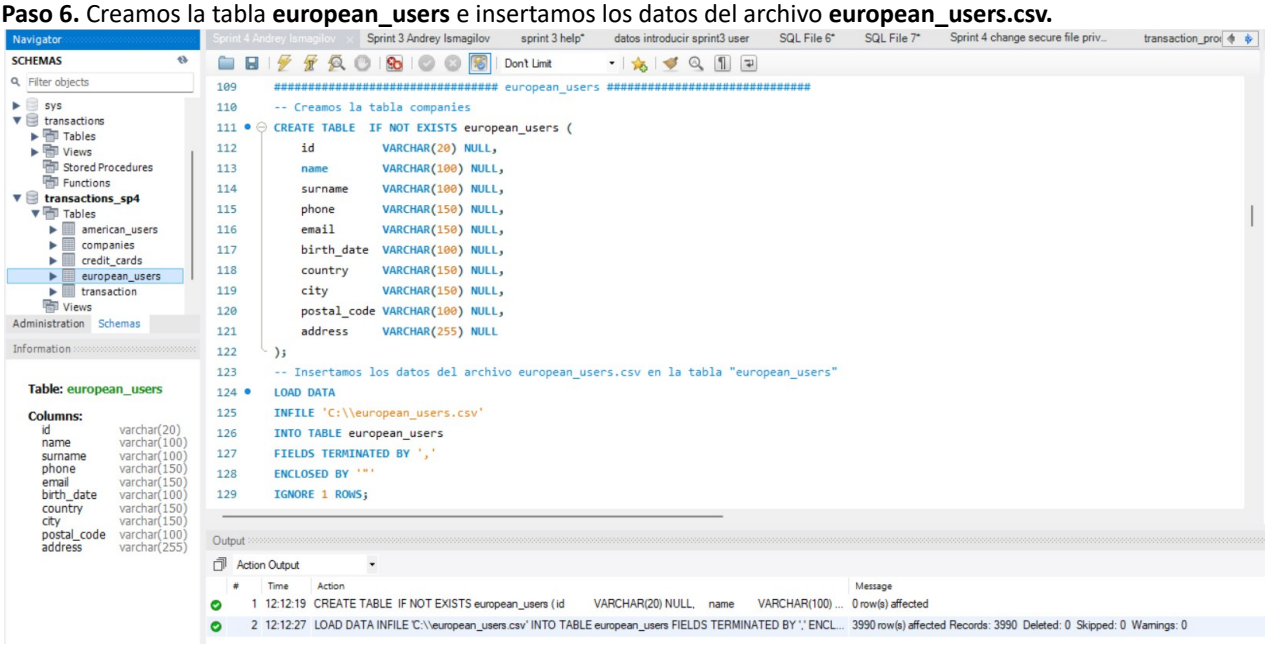
```
-- Creamos la tabla companies
CREATE TABLE  IF NOT EXISTS american_users (
    id          VARCHAR(20) NULL,
    name        VARCHAR(100) NULL,
    surname     VARCHAR(100) NULL,
    phone       VARCHAR(150) NULL,
    email       VARCHAR(150) NULL,
    birth_date  VARCHAR(100) NULL,
    country     VARCHAR(150) NULL,
    city        VARCHAR(150) NULL,
    postal_code VARCHAR(100) NULL,
    address     VARCHAR(255) NULL
);
-- Insertamos los datos del archivo american_users.csv en la tabla "american_users"
LOAD DATA
INFILE 'C:\\american_users.csv'
INTO TABLE american_users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 ROWS;
```

La **american_users** tiene 1010 registros.



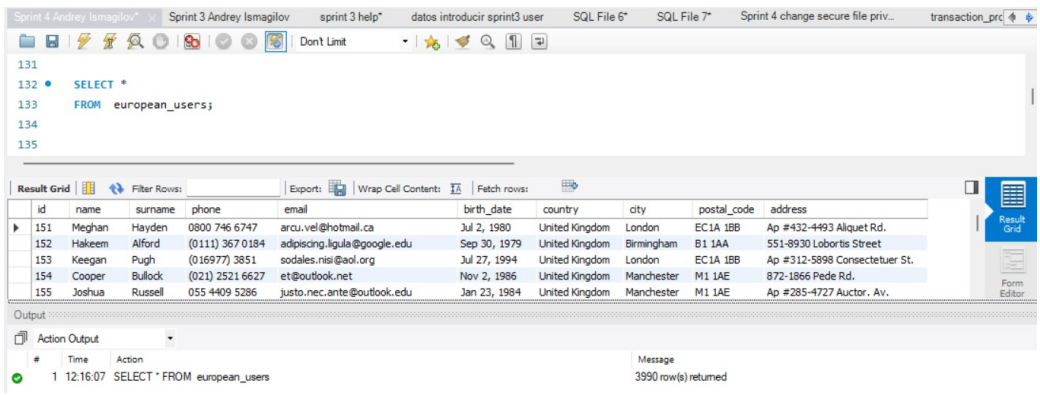
```
SELECT *
FROM american_users;
```

Paso 6. Creamos la tabla **european_users** e insertamos los datos del archivo **european_users.csv.**



```
-- Creamos la tabla companies
CREATE TABLE  IF NOT EXISTS european_users (
    id          VARCHAR(20) NULL,
    name        VARCHAR(100) NULL,
    surname     VARCHAR(100) NULL,
    phone       VARCHAR(150) NULL,
    email       VARCHAR(150) NULL,
    birth_date  VARCHAR(100) NULL,
    country     VARCHAR(150) NULL,
    city        VARCHAR(150) NULL,
    postal_code VARCHAR(100) NULL,
    address     VARCHAR(255) NULL
);
-- Insertamos los datos del archivo european_users.csv en la tabla "european_users"
LOAD DATA
INFILE 'C:\\european_users.csv'
INTO TABLE european_users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
IGNORE 1 ROWS;
```

La european_users tiene **3990** registros.

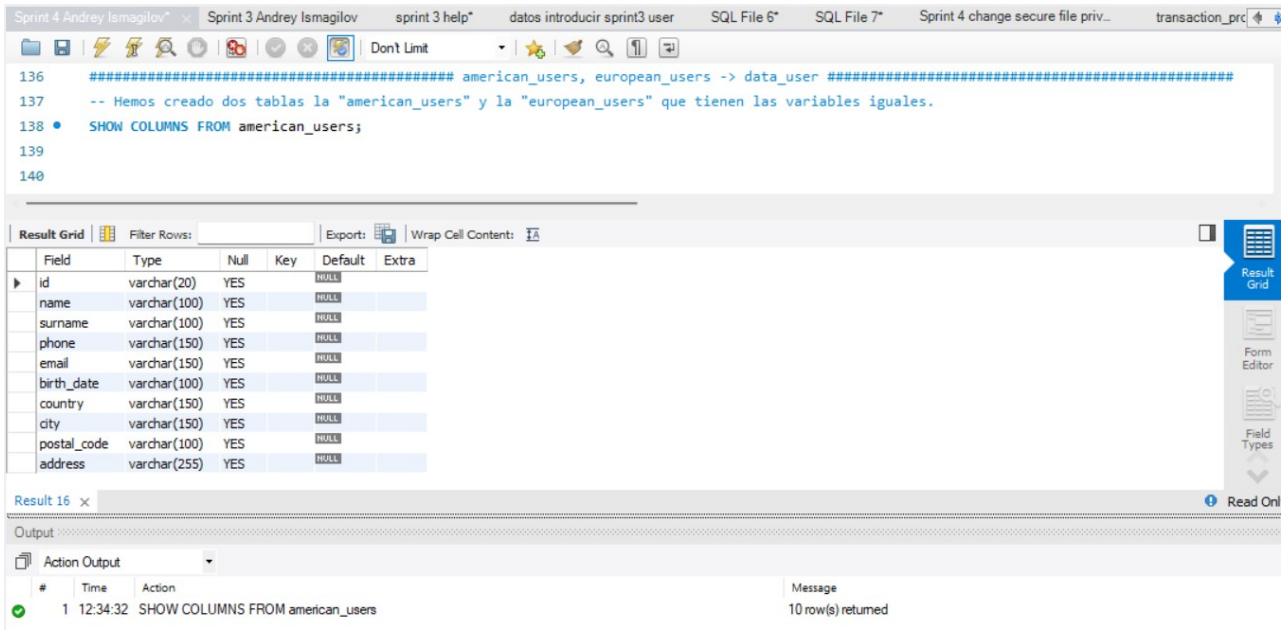


```
SELECT *
FROM european_users;
```

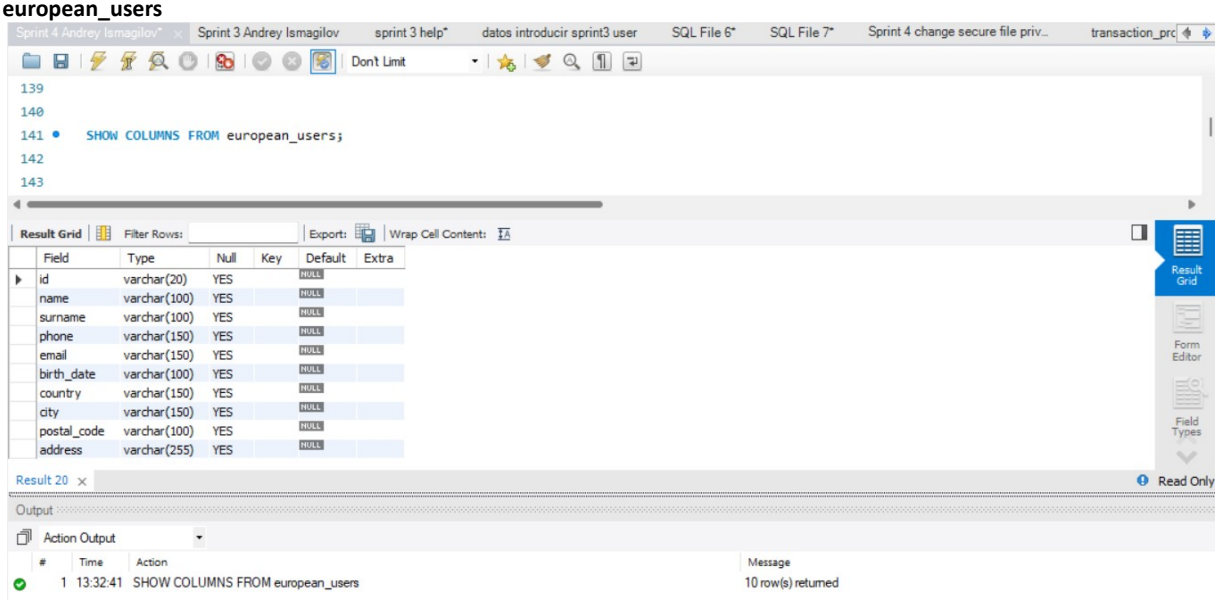
Paso 7. Creamos la tabla **data_user** combinando la tabla **american_users** y la **european_users.**

Antes de hacerlo miramos las columnas de ambas tablas. Vemos que las tienen las mismas columnas.

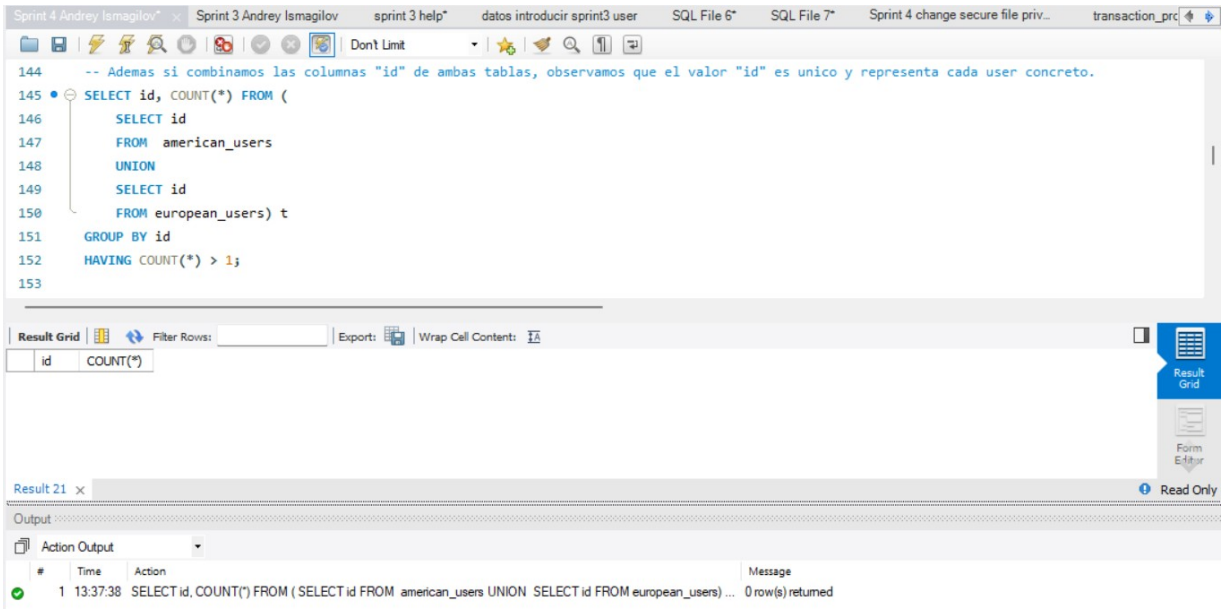
american_users



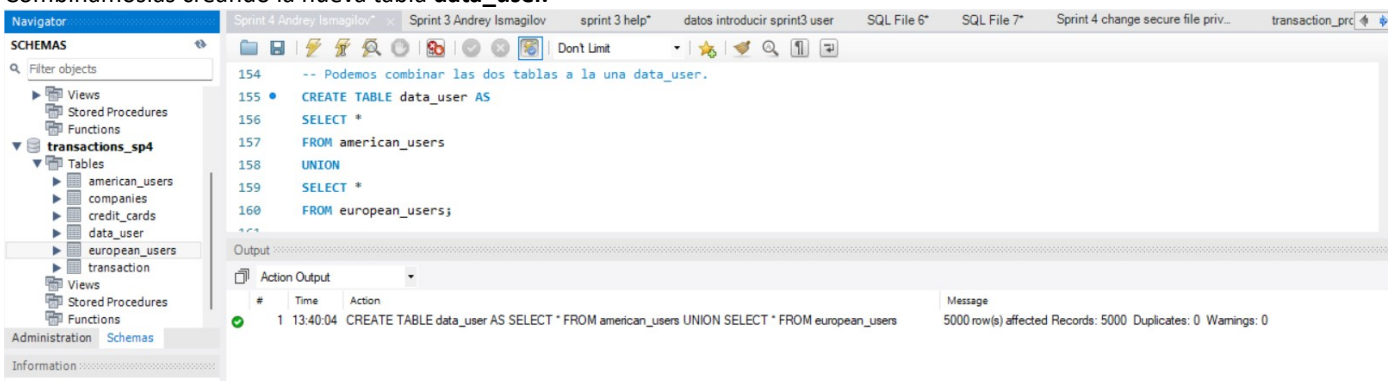
european_users



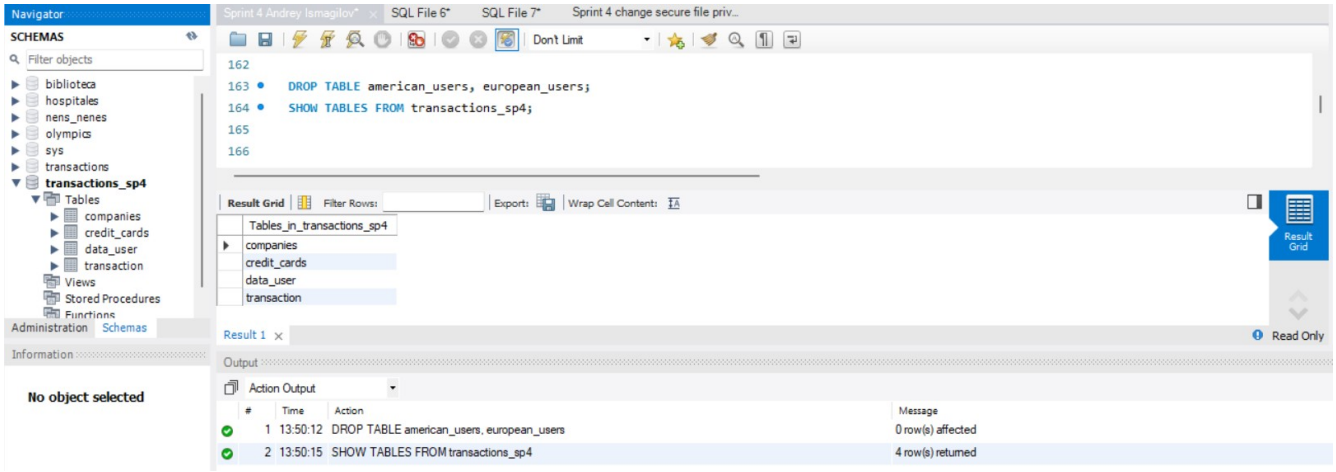
Vamos a comprobar que la columna combinada **id** tendra los valores únicos.



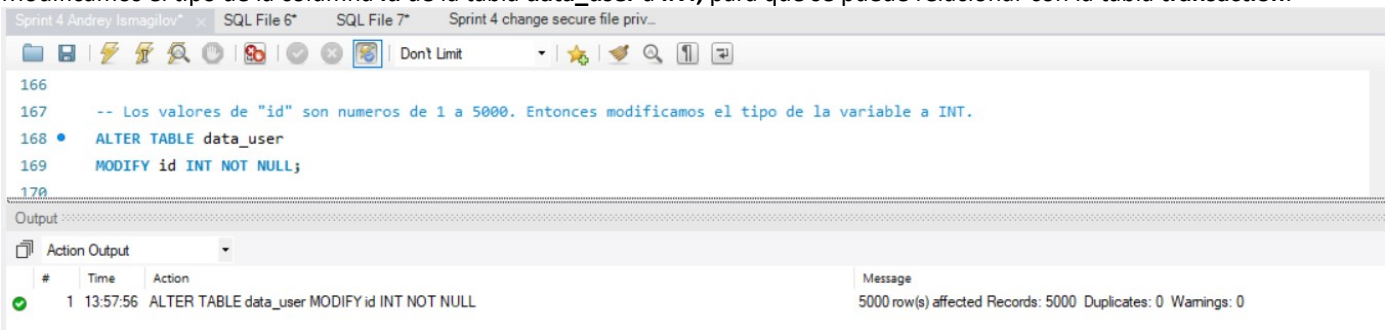
Combinamoslas creando la nueva tabla **data_user.**



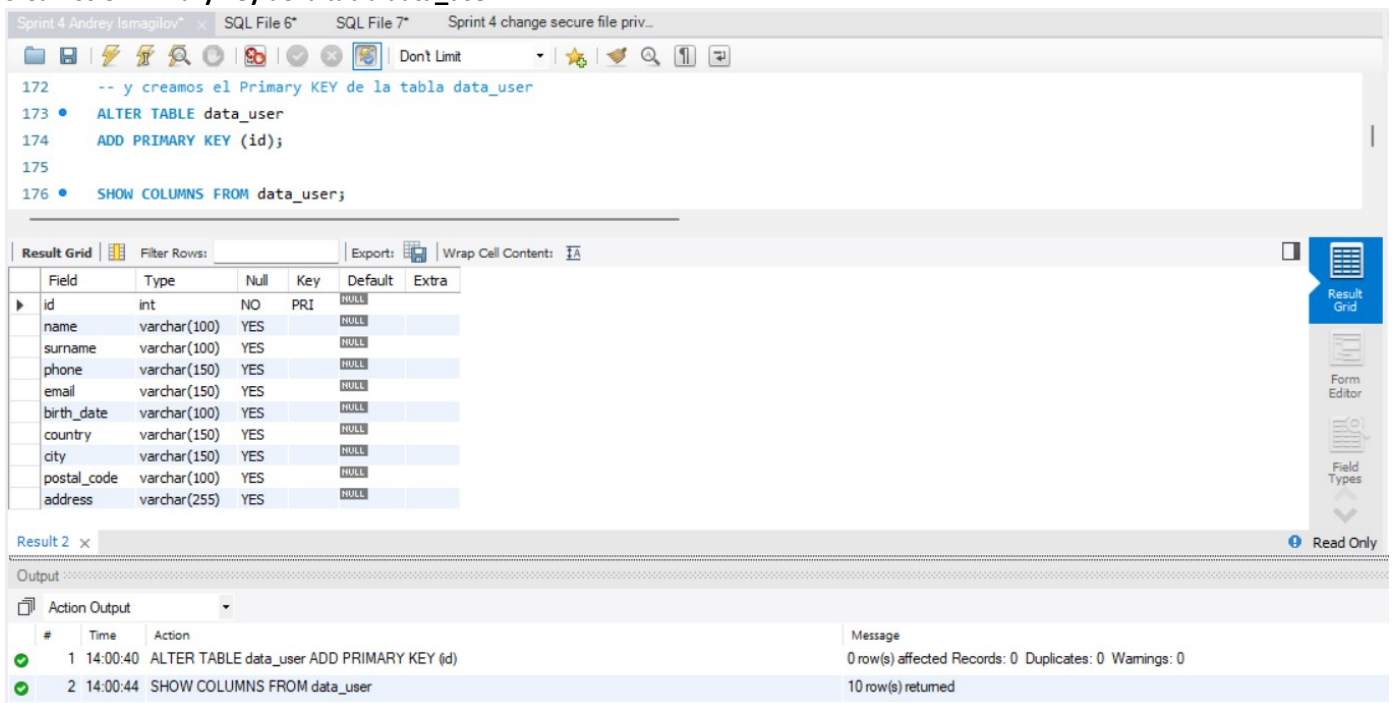
Eliminamos las tablas **american_users** y **european_users.**



Modificamos el tipo de la columna **id** de la tabla **data_user** a **INT,** para que se puede relacionar con la tabla **transaction**.

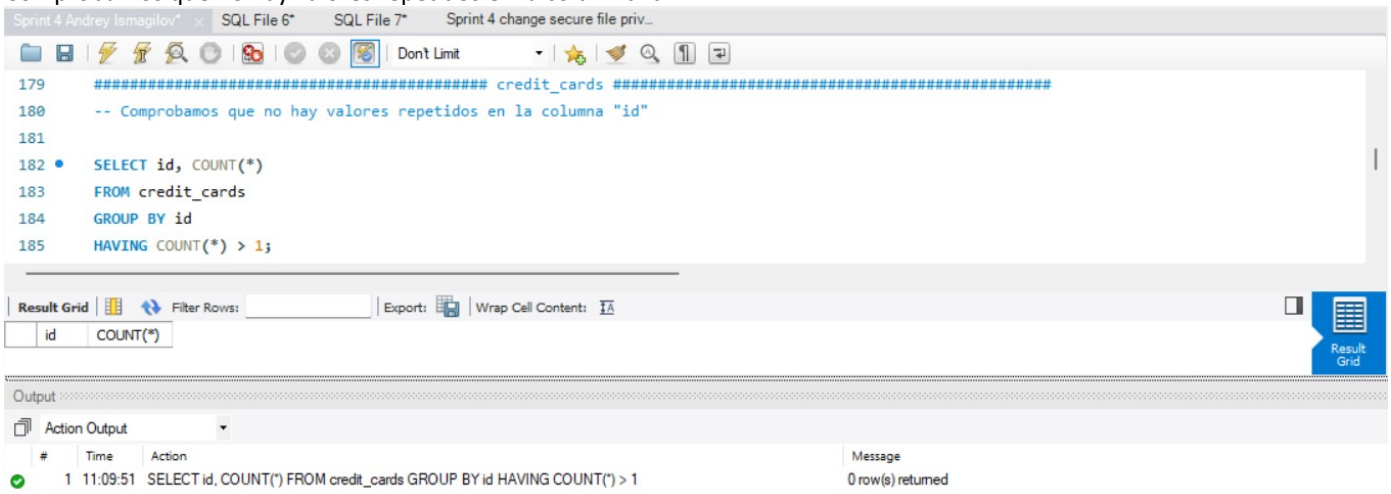


Creamos el **Primary Key** de la tabla **data_user.**

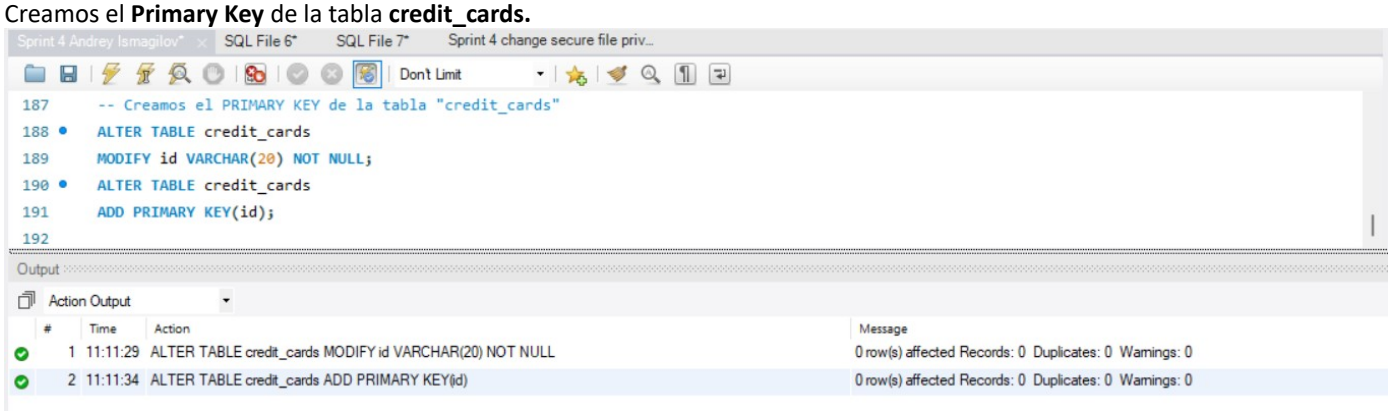


Paso 8. Modificamos la tabla **credit_cards**.

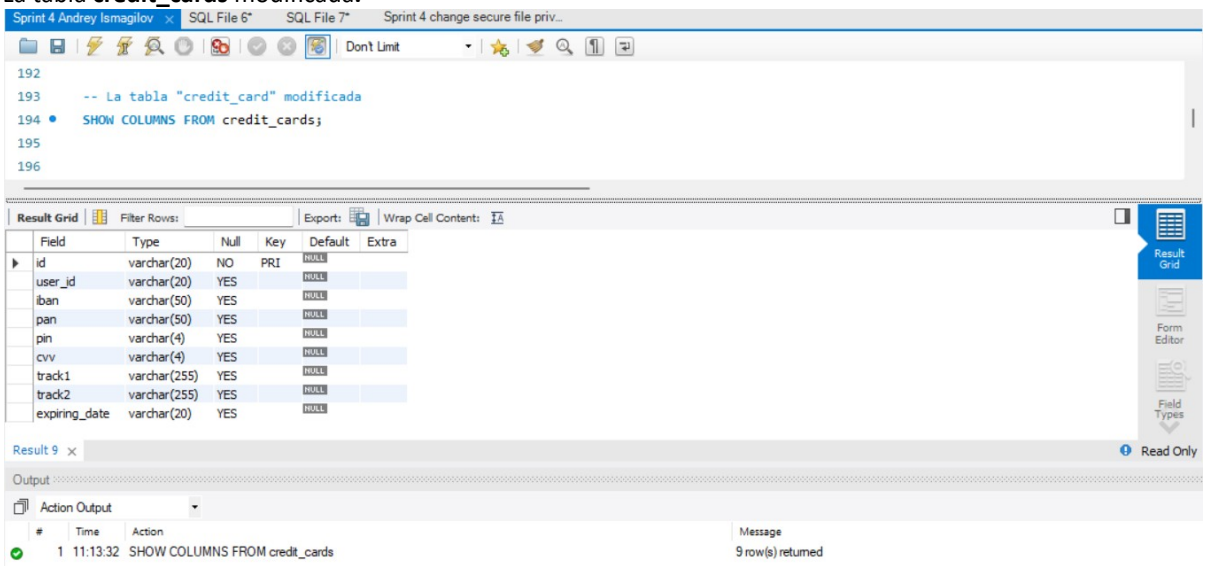
Comprobamos que no hay valores repetidos en la columna **id.**



Creamos el **Primary Key** de la tabla **credit_cards.**

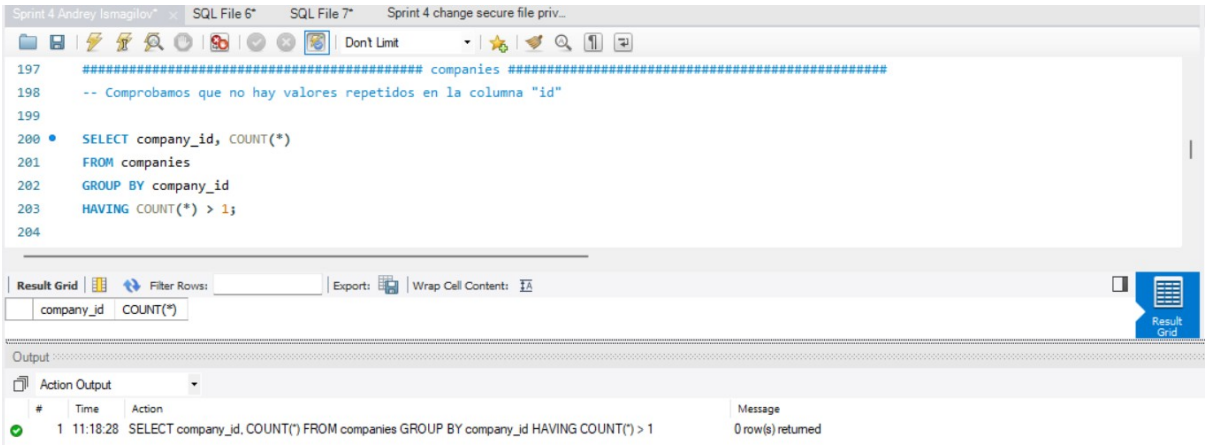


La tabla **credit_cards** modificada**.**

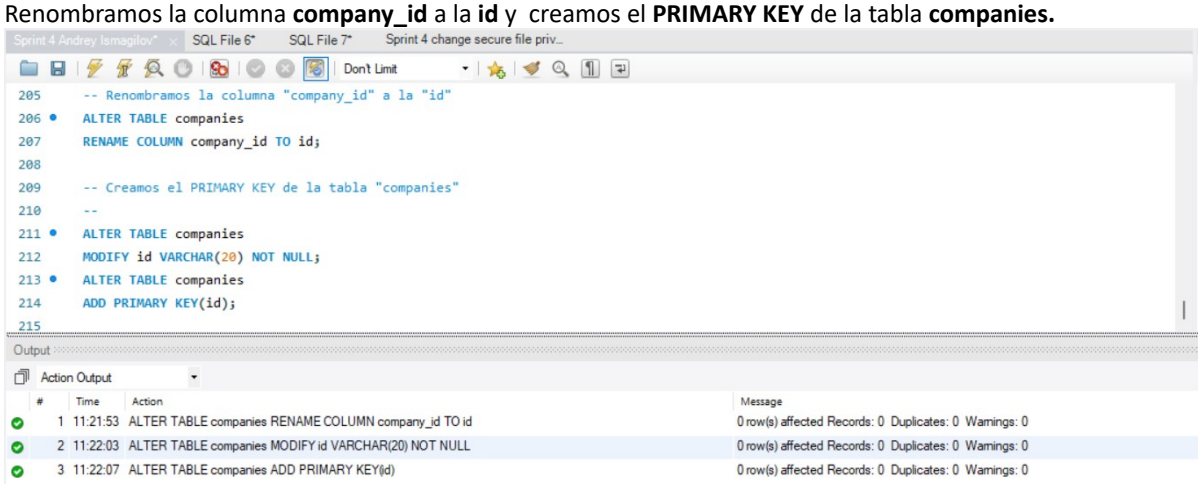


Paso 9. Modificamos la tabla **companies**.

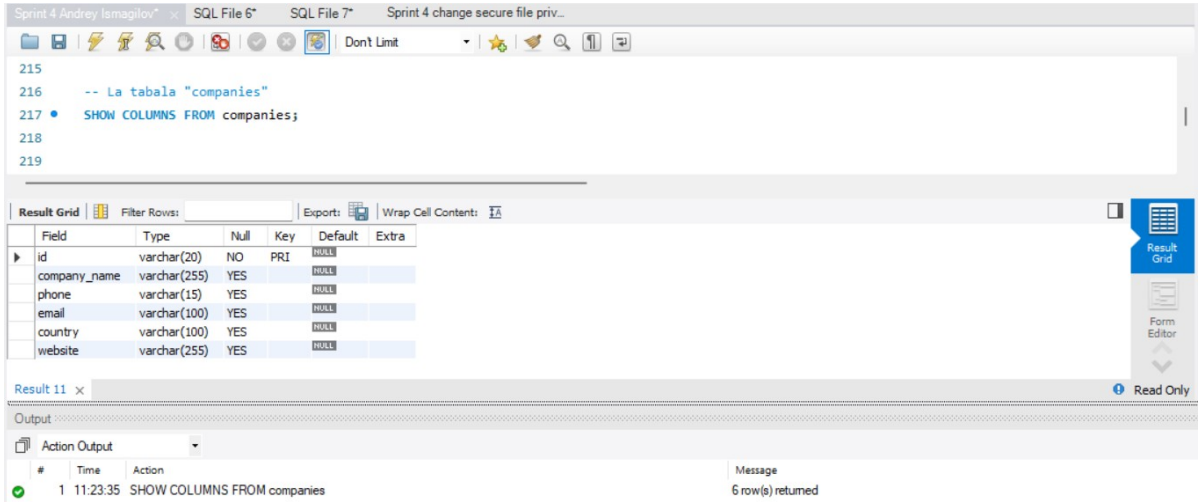
Comprobamos que no hay valores repetidos en la columna **id.**



Renombramos la columna **company_id** a la **id** y creamos el **PRIMARY KEY** de la tabla **companies.**

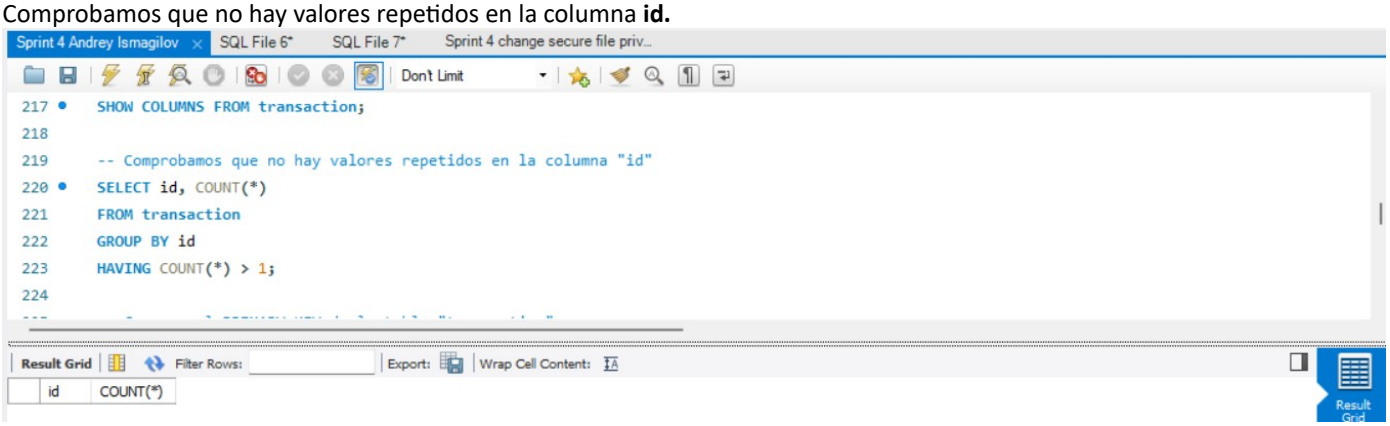


La tabla **companies** modificada.

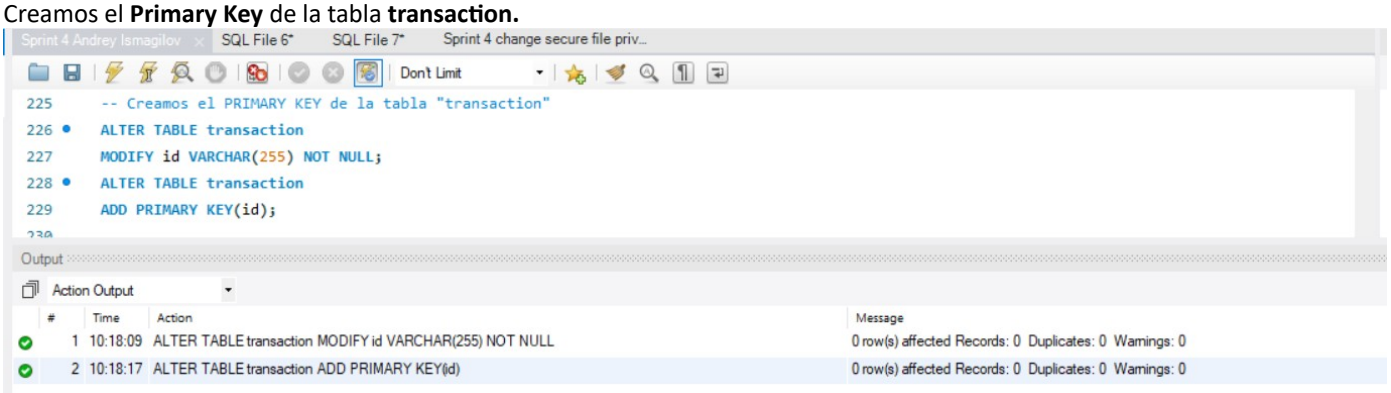


Paso 10. Modificamos la tabla **transaction.**

Comprobamos que no hay valores repetidos en la columna **id.**

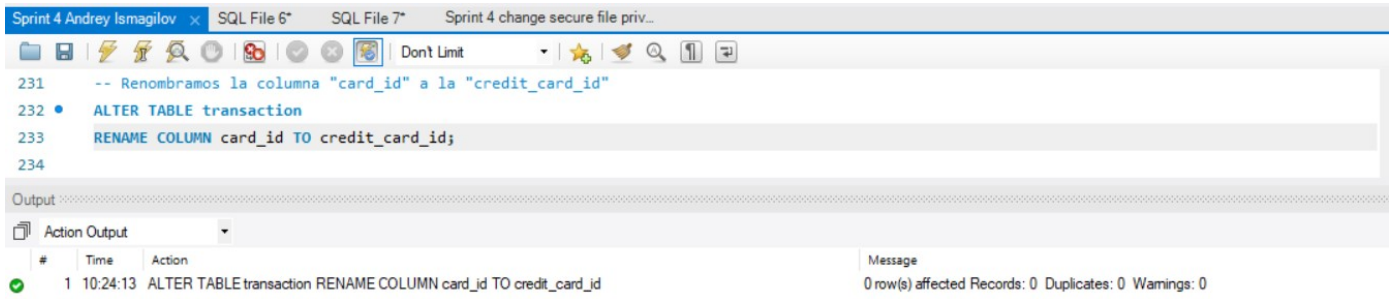


Creamos el **Primary Key** de la tabla **transaction.**

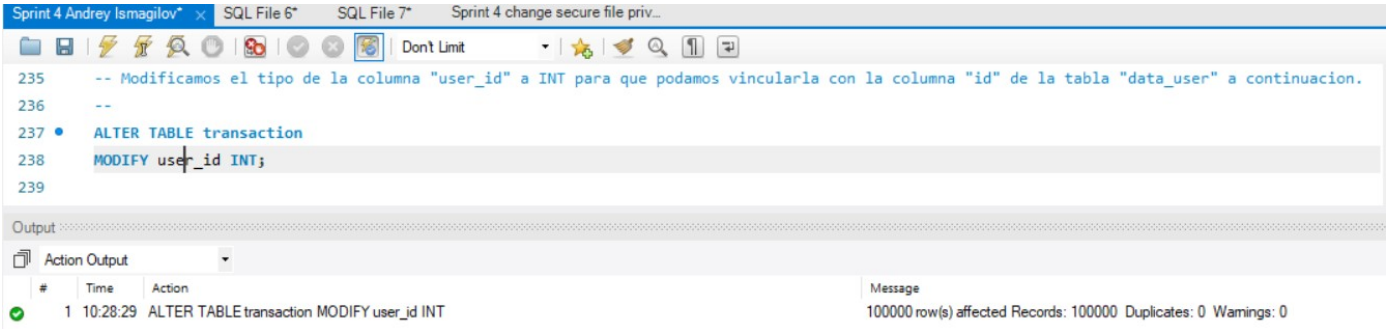


Renombramos la columna **card_id** y modificamos los tipos de varias columnas de la tabla **transaction**.

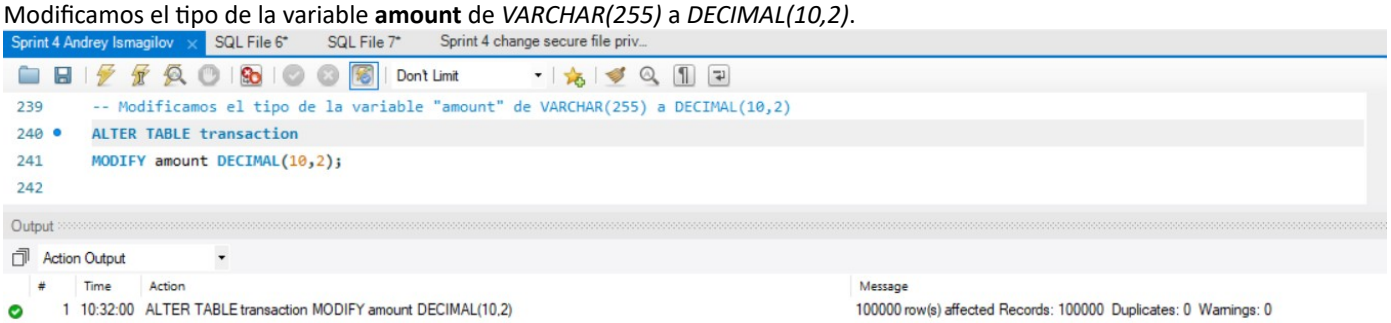
La **card_id** a la **credit_card_id**.



Modificamos el tipo de la columna **user_id** a *INT* para que podamos vincularla con la columna **id** de la tabla **data_user** a continuacion.



Modificamos el tipo de la variable **amount** de *VARCHAR(255)* a *DECIMAL(10,2)*.



Modificamos los tipos varias variables:

- **lat** de *VARCHAR(255)* a *FLOAT*.
- **longitude** de *VARCHAR(255)* a *FLOAT*.
- **declined** de *VARCHAR(4)* a **TINYINT(1)**.
- **timestamp** de *VARCHAR(255)* a *TIMESTAMP*.

```sql
-- Modificamos
-- el tipo de la variable "lat" de VARCHAR(255) a FLOAT,
-- el tipo de la variable "longitude" de VARCHAR(255) a FLOAT,
-- el tipo de la variable "declined" de VARCHAR(4) a TINYINT(1),
-- el tipo de la variable "timestamp" de VARCHAR(255) a TIMESTAMP.

ALTER TABLE transaction
MODIFY lat FLOAT,
MODIFY longitude FLOAT,
MODIFY declined TINYINT,
MODIFY timestamp TIMESTAMP;
```

```
Output
Action Output
#  Time      Action                                                                                                  Message
1  10:39:30  ALTER TABLE transaction MODIFY lat FLOAT, MODIFY longitude FLOAT, MODIFY declined TINYINT, MODIF...  0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
```

Al final tenemos la tabla **transaction** preparada para hacer la esquema de estrella.

```sql
-- Al final tenemos la tabla "transaction" preparada para hacel la esquema de estrella.
SHOW COLUMNS FROM transaction;
```

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

Field	Type	Null	Key	Default	Extra
id	varchar(255)	NO	PRI	NULL	
credit_card_id	varchar(20)	YES		NULL	
company_id	varchar(20)	YES		NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	YES		NULL	
declined	tinyint	YES		NULL	
product_ids	varchar(255)	YES		NULL	
user_id	int	YES		NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	

Result 6 | Read Only

```
Output
Action Output
#  Time      Action                                  Message
1  10:40:53  SHOW COLUMNS FROM transaction           10 row(s) returned
```

Paso 11. Creamos la esquema de estrella.

Vemos que la tabla "transaction" es la tabla de hechos del esquema de estrella.

- Contiene campos claves que se unen a las tablas de dimensiónes: credit_card_id, company_id, user_id.
- Contiene métricas que podemos medir y analizar: amount, timestamp, declined, lat, longitude.
- Cada fila de la tabla representa la transaccion unica de la tarjeta credito.

Vinculamos la columna **id** de la tabla **data_user** con la **user_id** de la "**transaction.**

```sql
########################################## el esquema de estrella ################################################
-- Creamos la esquema de estrella.
-- Vemos que la tabla "transaction" es la tabla de hechos del esquema de estrella.
-- Contiene campos claves que se unen a las tablas de dimensiónes: credit_card_id, company_id, user_id.
-- Contiene métricas que podemos media y analizar: amount, timestamp, declined, lat, longitude.
-- Cada fila de la tabla representa la transaccion unica de la tarjeta credito.

-- Vinculamos la columna "id" de la tabla "data_user" con la "user_id" de la "transaction"
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_user_id
FOREIGN KEY (user_id)
REFERENCES data_user (id);
```

```
Output
Action Output
#  Time      Action                                                                                                  Message
1  10:51:52  ALTER TABLE transaction ADD CONSTRAINT fk_transaction_user_id FOREIGN KEY (user_id) REFERENCES ...  100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
```

Vinculamos la columna **id** de la tabla **companies** con la **company_id** de la **transaction**.

```sql
-- Vinculamos la columna "id" de la tabla "companies" con la "company_id" de la "transaction"
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_company_id
FOREIGN KEY (company_id)
REFERENCES companies (id);
```

Output

#	Time	Action	Message
1	11:44:05	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_company_id FOREIGN KEY (company_id) REFE...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Vinculamos la columna **id** de la tabla **credit_cards** con la **credit_card_id** de la **transaction.**

```sql
-- Vinculamos la columna "id" de la tabla "credit_cards" con la "credit_card_id" de la "transaction"
ALTER TABLE transaction
ADD CONSTRAINT fk_transaction_credit_card_id
FOREIGN KEY (credit_card_id)
REFERENCES credit_cards (id);
```

Output

#	Time	Action	Message
1	11:48:02	ALTER TABLE transaction ADD CONSTRAINT fk_transaction_credit_card_id FOREIGN KEY (credit_card_id) R...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Vamos a mirar los **FOREIGN KEYS** creados.

```sql
-- Miramos los FOREIGN KEYS creados.
SELECT
    table_name,
    column_name,
    constraint_name,
    referenced_table_name,
    referenced_column_name
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
referenced_table_schema = 'transactions_sp4' ;
```

Result Grid

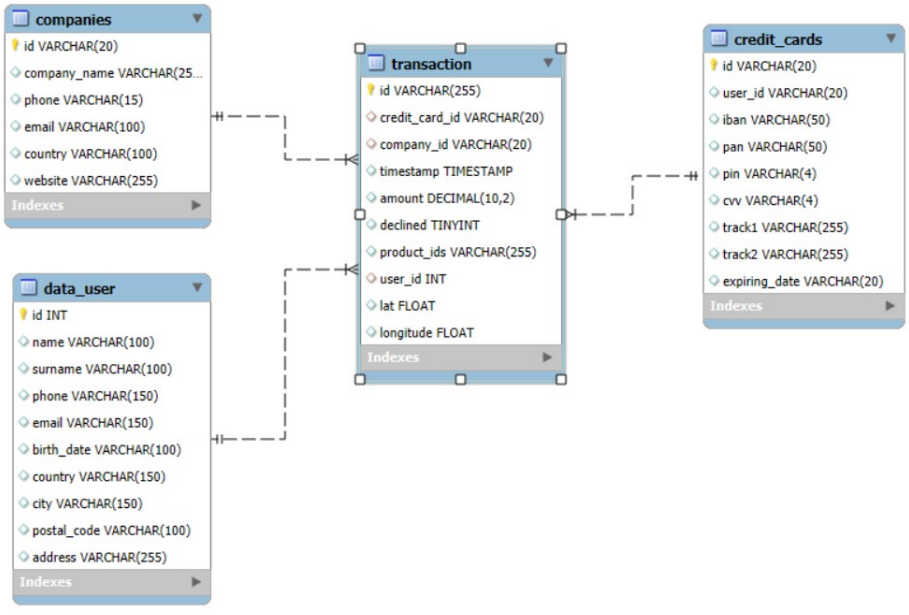
TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
transaction	company_id	fk_transaction_company_id	companies	id
transaction	credit_card_id	fk_transaction_credit_card_id	credit_cards	id
transaction	user_id	fk_transaction_user_id	data_user	id

KEY_COLUMN_USAGE 12 — Read Only

Output

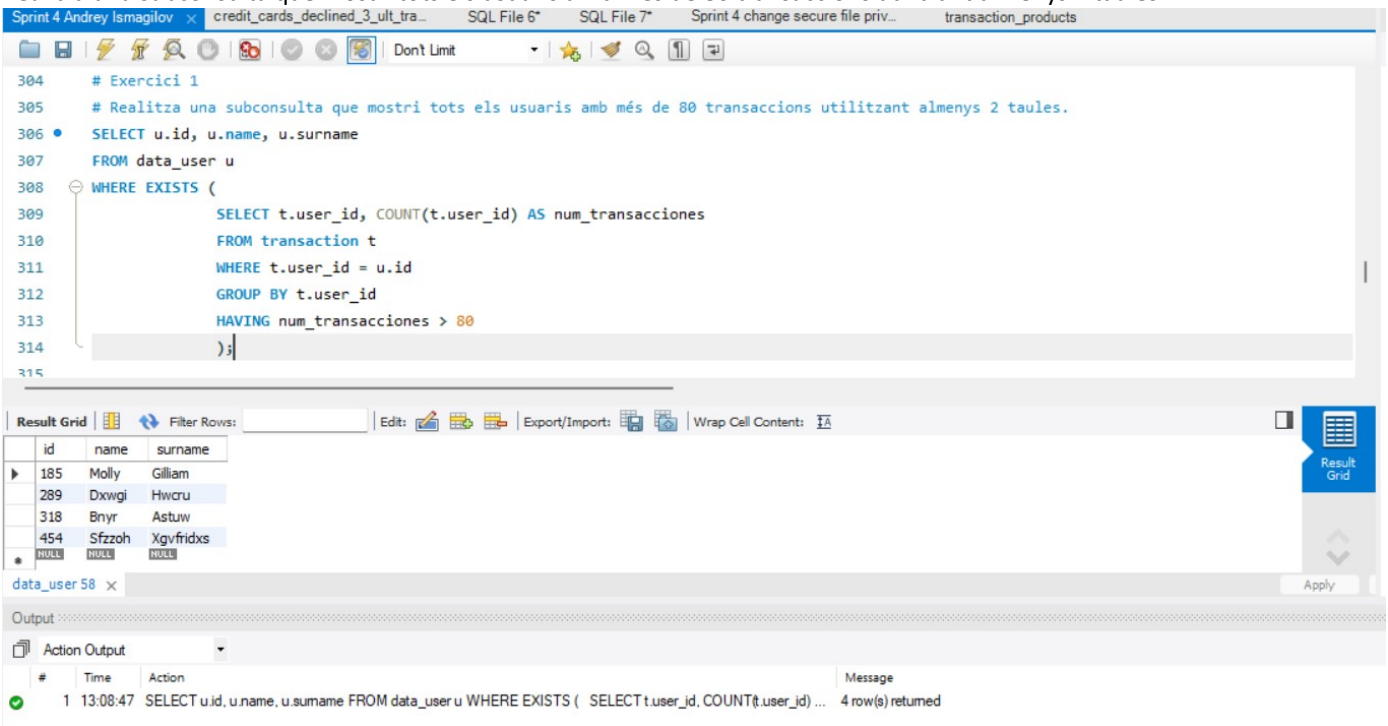
#	Time	Action	Message
1	11:49:33	SELECT table_name, column_name, constraint_name, referenced_table_name, referenced_colu...	3 row(s) returned

Al final hemos creado la esquema estrella que se representa en el siguiente diagrama de la base de datos **transaction_sp4**.



Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.

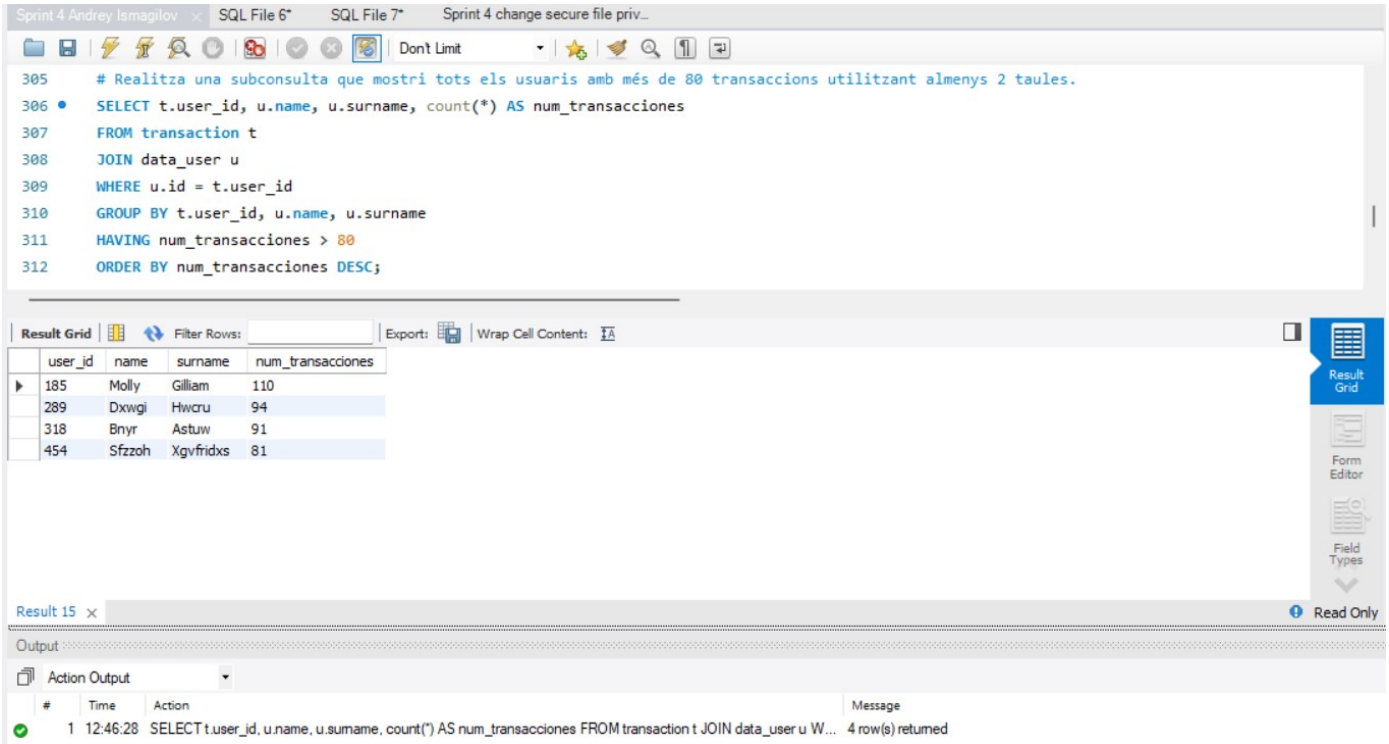


```
# Exercici 1
# Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.
SELECT u.id, u.name, u.surname
FROM data_user u
WHERE EXISTS (
            SELECT t.user_id, COUNT(t.user_id) AS num_transacciones
            FROM transaction t
            WHERE t.user_id = u.id
            GROUP BY t.user_id
            HAVING num_transacciones > 80
            );
```

id	name	surname
185	Molly	Gilliam
289	Dxwgi	Hwcru
318	Bnyr	Astuw
454	Sfzzoh	Xgvfridxs

```
Output
Action Output
#  Time      Action                                                                                                  Message
1  13:08:47  SELECT u.id, u.name, u.surname FROM data_user u WHERE EXISTS (  SELECT t.user_id, COUNT(t.user_id) ...  4 row(s) returned
```

Lo he hecho con un JOIN tambien tan solo para comparar los resultados.



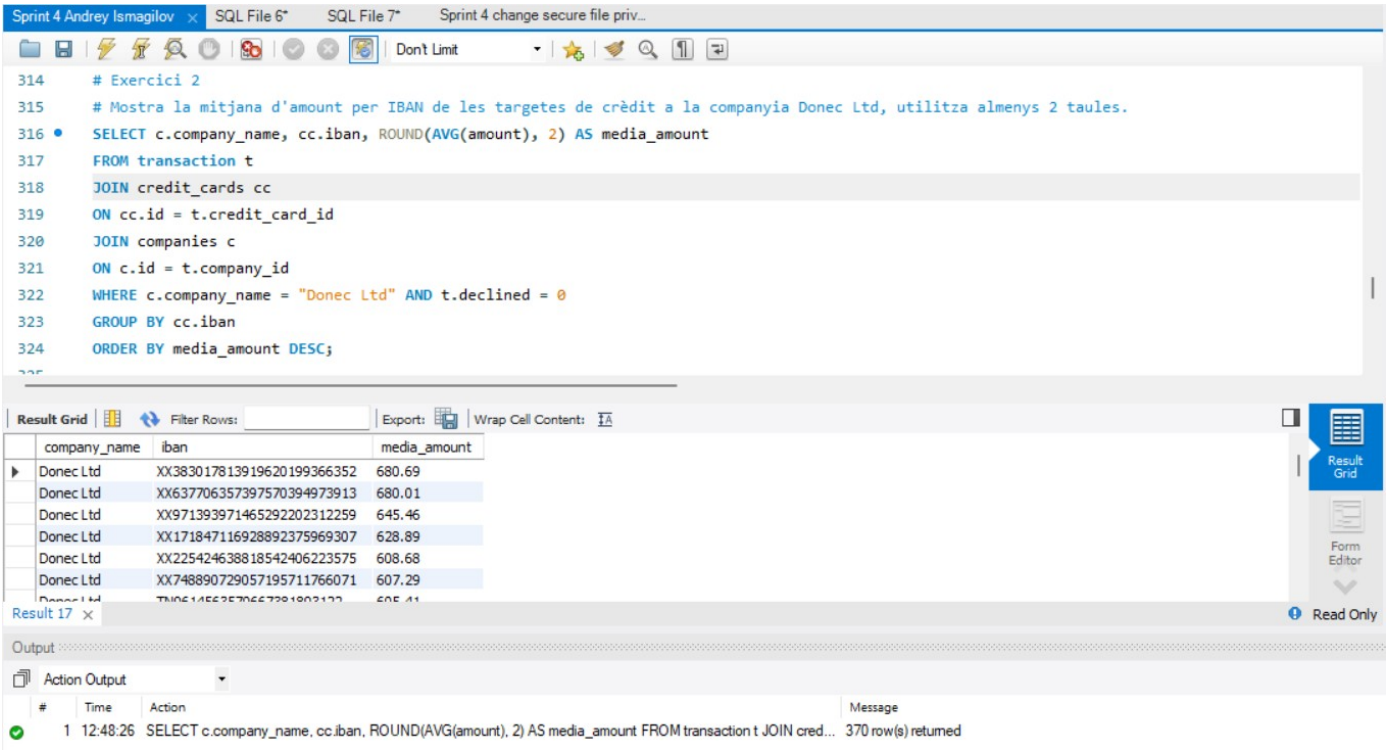
```
# Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.
SELECT t.user_id, u.name, u.surname, count(*) AS num_transacciones
FROM transaction t
JOIN data_user u
WHERE u.id = t.user_id
GROUP BY t.user_id, u.name, u.surname
HAVING num_transacciones > 80
ORDER BY num_transacciones DESC;
```

user_id	name	surname	num_transacciones
185	Molly	Gilliam	110
289	Dxwgi	Hwcru	94
318	Bnyr	Astuw	91
454	Sfzzoh	Xgvfridxs	81

```
Output
Action Output
#  Time      Action                                                                                                  Message
1  12:46:28  SELECT t.user_id, u.name, u.surname, count(*) AS num_transacciones FROM transaction t JOIN data_user u W...  4 row(s) returned
```

Exercici 2

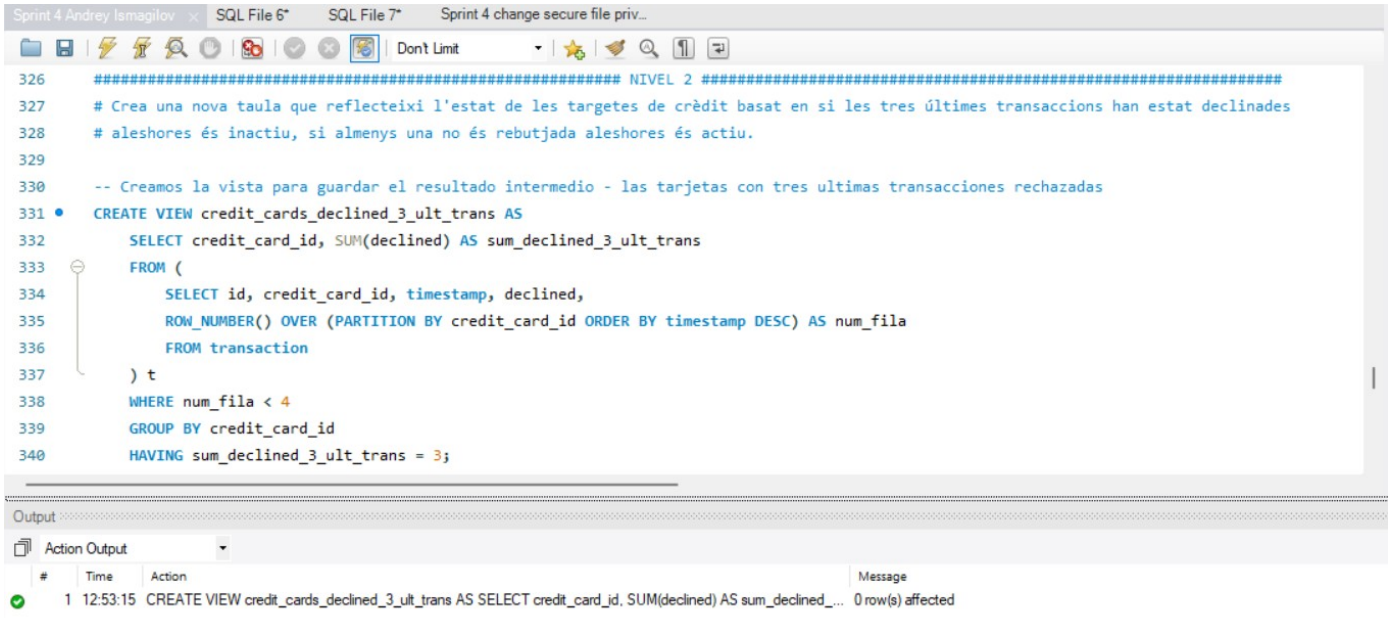
Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.



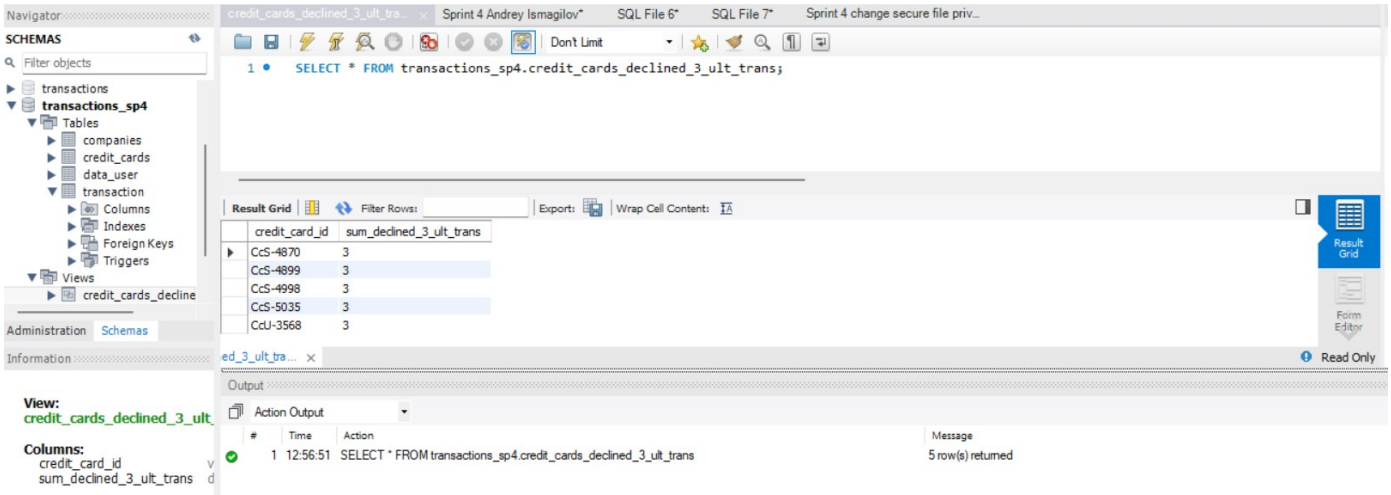
Nivel 2

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les tres últimes transaccions han estat declinades aleshores és inactiu, si almenys una no és rebutjada aleshores és actiu.

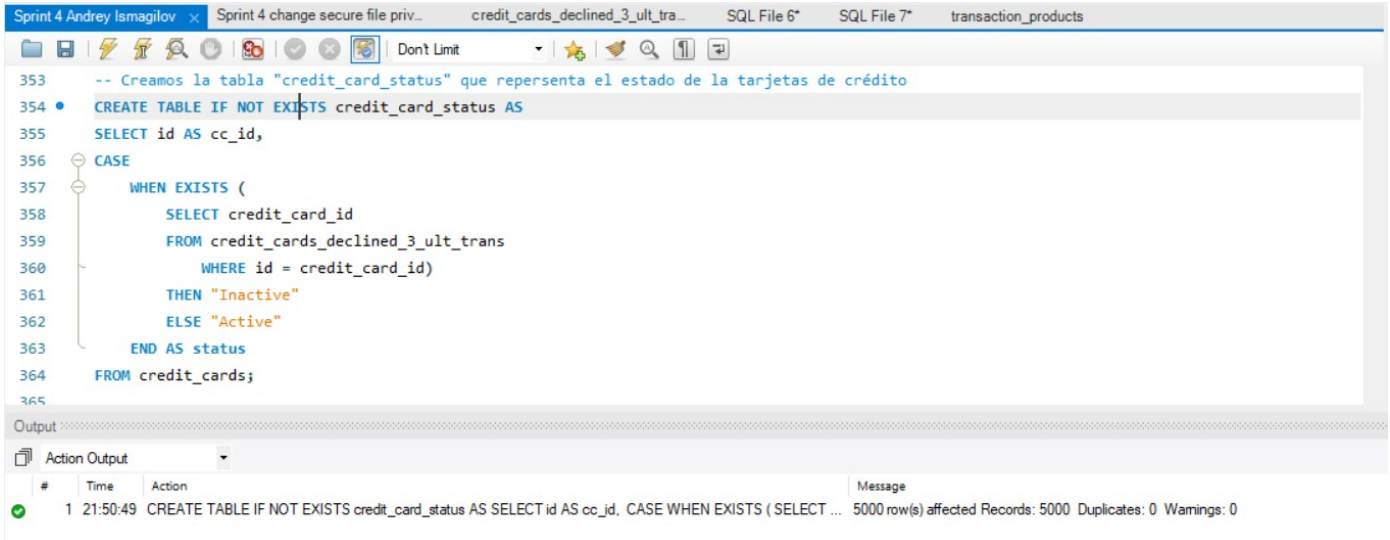
Creamos la vista **credit_cards_declinded_3_ult_trans** para guardar el resultado intermedio - las tarjetas con tres ultimas transacciones rechazadas



La vista **credit_cards_declinded_3_ult_trans.**



Creamos la tabla **credit_card_status** que repersenta el estado de la tarjetas de crédito.



La tabla **credit_card_status.**

```
SELECT *
FROM credit_card_status
ORDER BY status DESC;
```

cc_id	status
CcS-4998	Inactive
CcS-4870	Inactive
CcS-5035	Inactive
CcS-4899	Inactive
CcU-3568	Inactive
CcS-4858	Active
CcS-4859	Active

Output

#	Time	Action	Message
1	21:52:34	SELECT * FROM credit_card_status ORDER BY status DESC	5000 row(s) returned

Vinculamos la columna **id** de la tabla **credit_cards** con la **cc_id** de la tabla **credit_card_status**.

```
-- Vinculamos la columna "id" de la tabla "credit_cards" con la cc_id de la tabla "credit_card_status".
ALTER TABLE credit_cards
ADD CONSTRAINT fk_credit_cards_id
FOREIGN KEY (id)
REFERENCES credit_card_status(cc_id);
-- Miramos los FOREIGN KEYS creados.
SELECT
    table_name,
    column_name,
    constraint_name,
    referenced_table_name,
    referenced_column_name
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
referenced_table_schema = 'transactions_sp4' ;
```

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
credit_cards	id	fk_credit_cards_id	credit_card_status	cc_id
transaction	company_id	fk_transaction_company_id	companies	id
transaction	credit_card_id	fk_transaction_credit_card_id	credit_cards	id
transaction	user_id	fk_transaction_user_id	data_user	id

Output

#	Time	Action	Message
1	19:09:05	ALTER TABLE credit_cards ADD CONSTRAINT fk_credit_cards_id FOREIGN KEY (id) REFERENCES credit_c...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
2	19:09:08	SELECT table_name, column_name, constraint_name, referenced_table_name, referenced_colu...	4 row(s) returned

Exercici 1

Quantes targetes estan actives?

```
# Exercici 1
#Quantes targetes estan actives?
SELECT COUNT(*) AS num_tarjetas_activas
FROM credit_card_status
WHERE status = "Active";
```

num_tarjetas_activas
4995

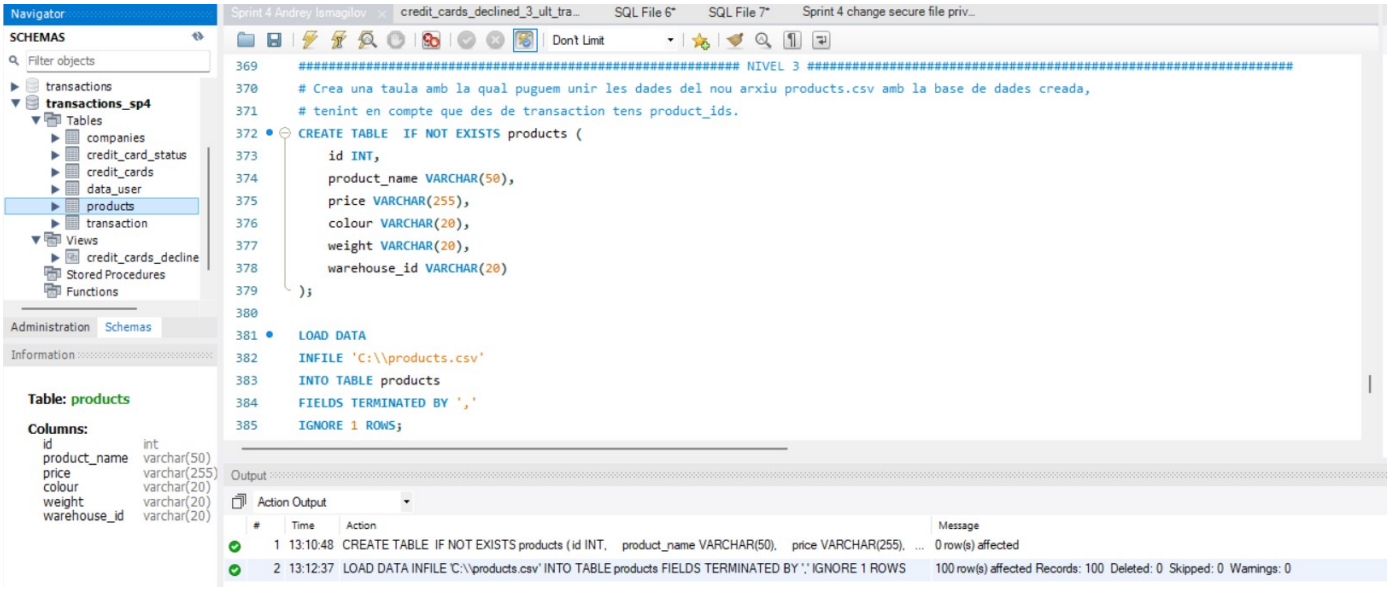
Output

#	Time	Action	Message
1	13:06:29	SELECT COUNT(*) AS num_tarjetas_activas FROM credit_card_status WHERE status = "Active"	1 row(s) returned

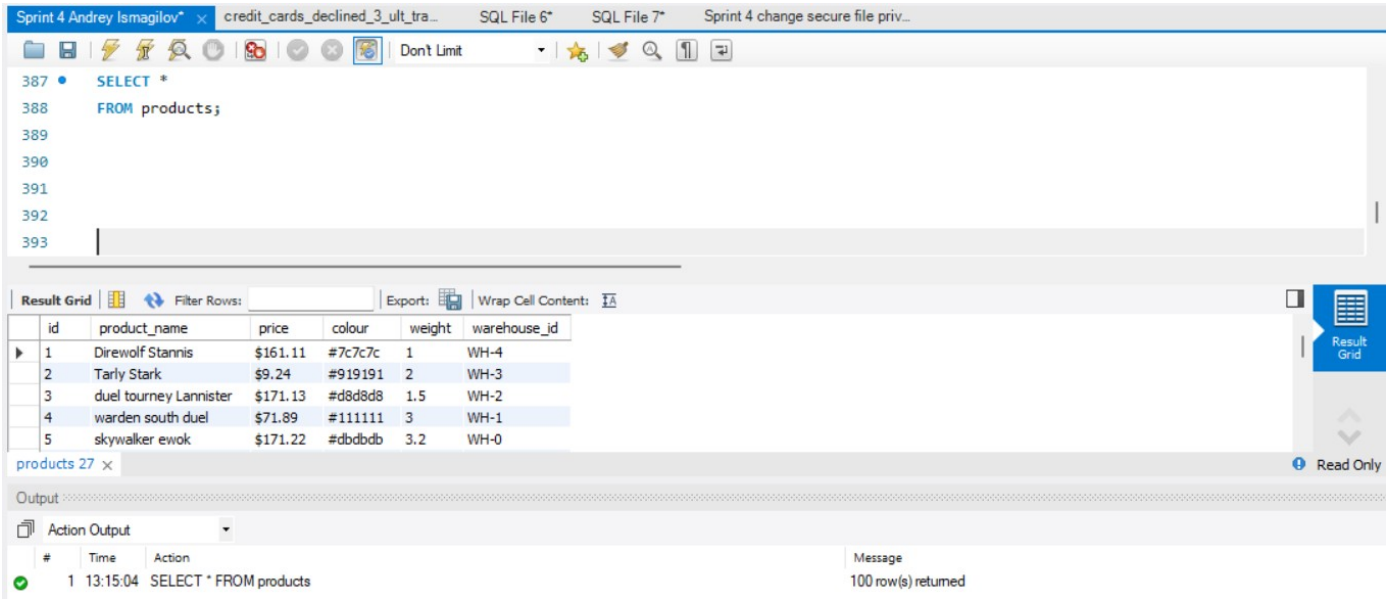
Nivel 3

Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids.

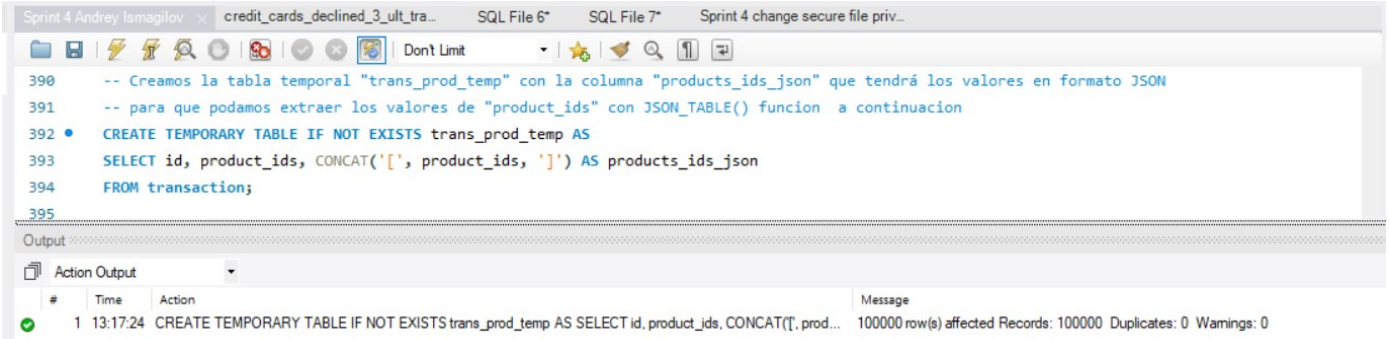
Creamos la tabla **products** y insertamos los datos del archivo **products.csv** en la tabla**.**



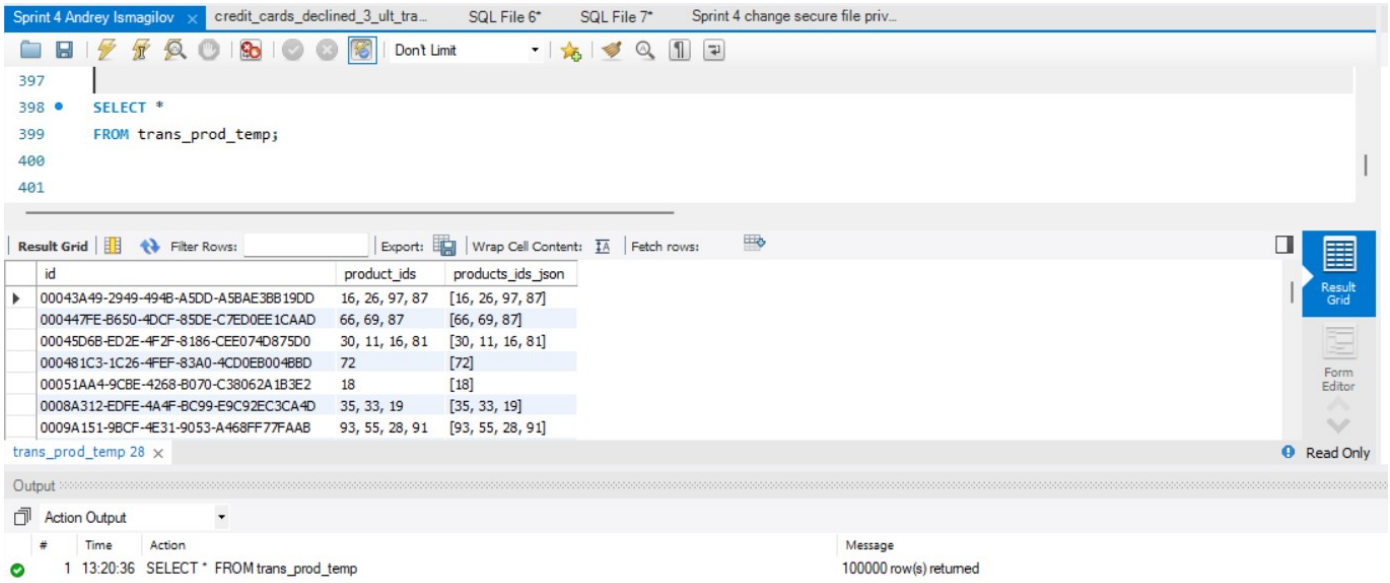
La tabla **products.**



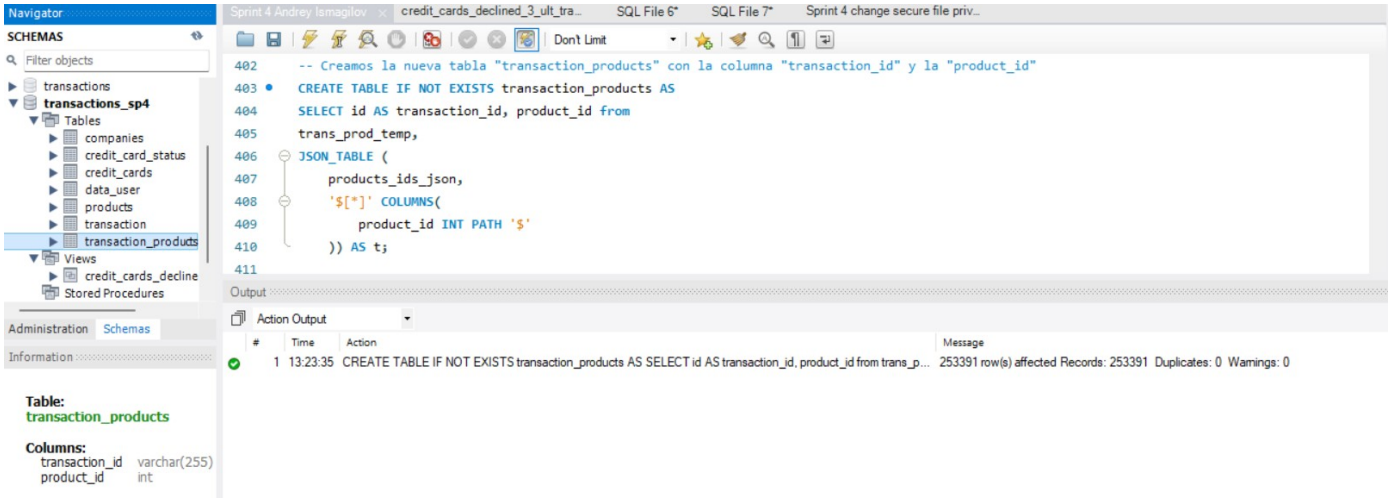
Creamos la tabla temporal **trans_prod_temp** con la columna **products_ids_json** que tendrá los valores en formato JSON para que podamos extraer los valores de **product_ids** con JSON_TABLE() función a continuacion.



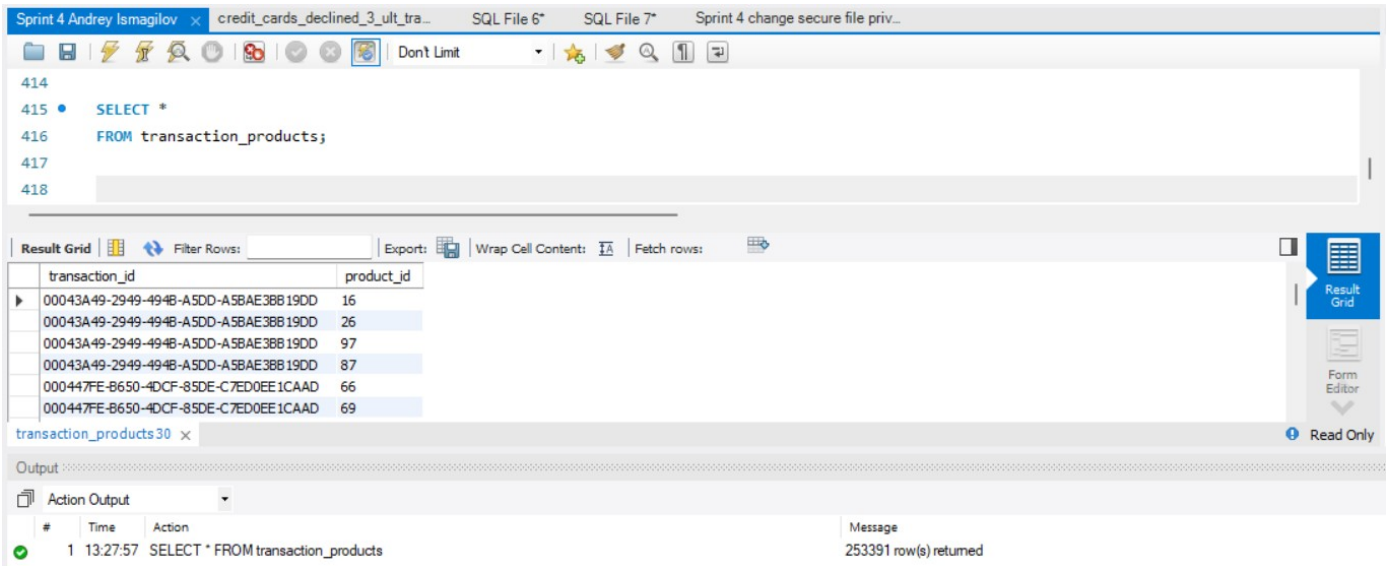
La tabla temporal **trans_prod_temp.**



Creamos la nueva tabla **transaction_products** con la columna **transaction_id** y la **product_id**.



La tabla **transaction_products.**



Unimos la tabla **products** con las tablas creadas utilizando la tabla **transaction_products** como la tabla-puente.
Para ello vinculamos la columna **transaction_id** de la tabla **transaction_products** con la **id** de la **transaction** y luego vinculamos la columna "id" de la **products** con la columna **product_id** de la **transaction_products**.

```sql
-- Vinculamos la columna "transaction_id" de la tabla "transaction_products" con la "id" de la tabla "transaction".
ALTER TABLE transaction_products
ADD CONSTRAINT fk_transaction_products_transaction_id
FOREIGN KEY(transaction_id)
REFERENCES transaction(id);

-- Vinculamos la columna "id" de la tabla "products" con la columna "product_id" de la tabla "transaction_products".
ALTER TABLE products
ADD PRIMARY KEY(id);

ALTER TABLE transaction_products
ADD CONSTRAINT fk_transaction_products_product_id
FOREIGN KEY(product_id)
REFERENCES products(id);
```

Output

Action Output

#	Time	Action	Message
1	21:00:56	ALTER TABLE transaction_products ADD CONSTRAINT fk_transaction_products_transaction_id FOREIGN KEY...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0
2	21:01:02	ALTER TABLE products ADD PRIMARY KEY(id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
3	21:01:09	ALTER TABLE transaction_products ADD CONSTRAINT fk_transaction_products_product_id FOREIGN KEY(pr...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

Miramos los **FOREIGN KEYS** creadas.

```sql
-- Miramos los FOREIGN KEYS creados.
SELECT
    table_name,
    column_name,
    constraint_name,
    referenced_table_name,
    referenced_column_name
FROM
    INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE
referenced_table_schema = 'transactions_sp4'  AND table_name = 'transaction_products';
```

Result Grid

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
transaction_products	product_id	fk_transaction_products_product_id	products	id
transaction_products	transaction_id	fk_transaction_products_transaction_id	transaction	id

fk_transaction_products_transaction_id

KEY_COLUMN_USAGE 84

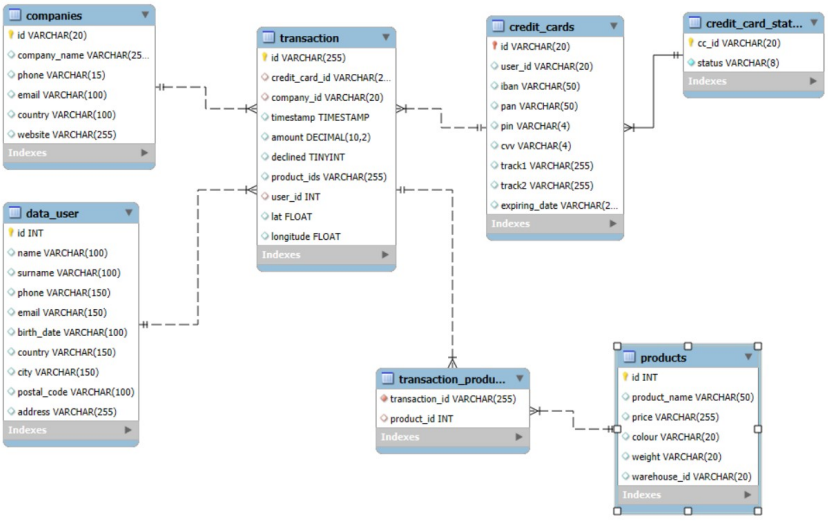
Read Only

Output

Action Output

#	Time	Action	Message
1	21:05:00	SELECT table_name, column_name, constraint_name, referenced_table_name, referenced_colu...	2 row(s) returned

El diagrama final de la esquema entitad-relacion con todas las tablas creadas.



Exercici 1

Necessitem conèixer el nombre de vegades que s'ha venut cada producte.