**S4.01. Creación de Base de Datos**

**Creación de DB y adecuación**

Primero creamos la base de datos (a la cual llamaremos bussines) y las tablas correspondientes a los archivos CSV proporcionados.

Usaremos el comando CREATE TABLE, al cual añadiremos IF NOT EXISTS para que cree las tablas solo en caso de que no existan.

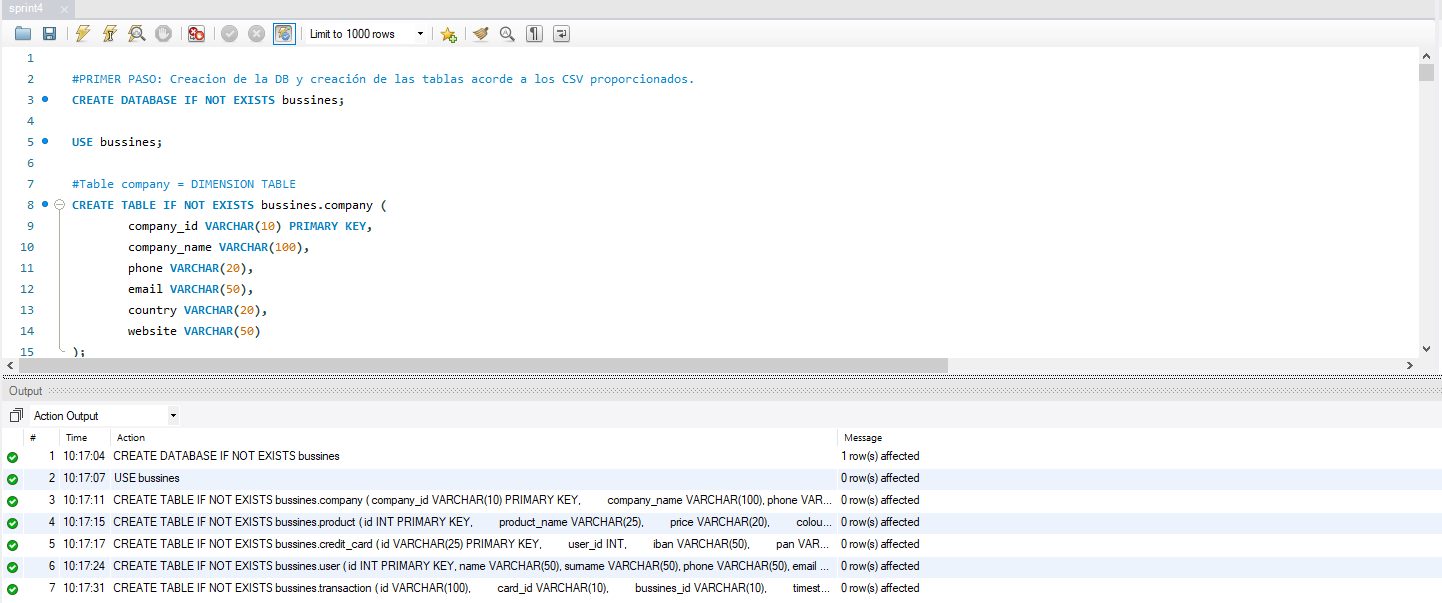
Según los archivos proporcionados tendremos un modelo en estrella con la tabla de *transaction* como **FACT TABLE**, y las tablas *user, product, credit\_card y company* como **DIMENSION TABLE**.

Los archivos CSV se podrían cargar o bien mediante comandos o mediante el *import wizard* del *MySQL Workbench*. En este caso usaremos el *import wizard.*

Creación de tablas:

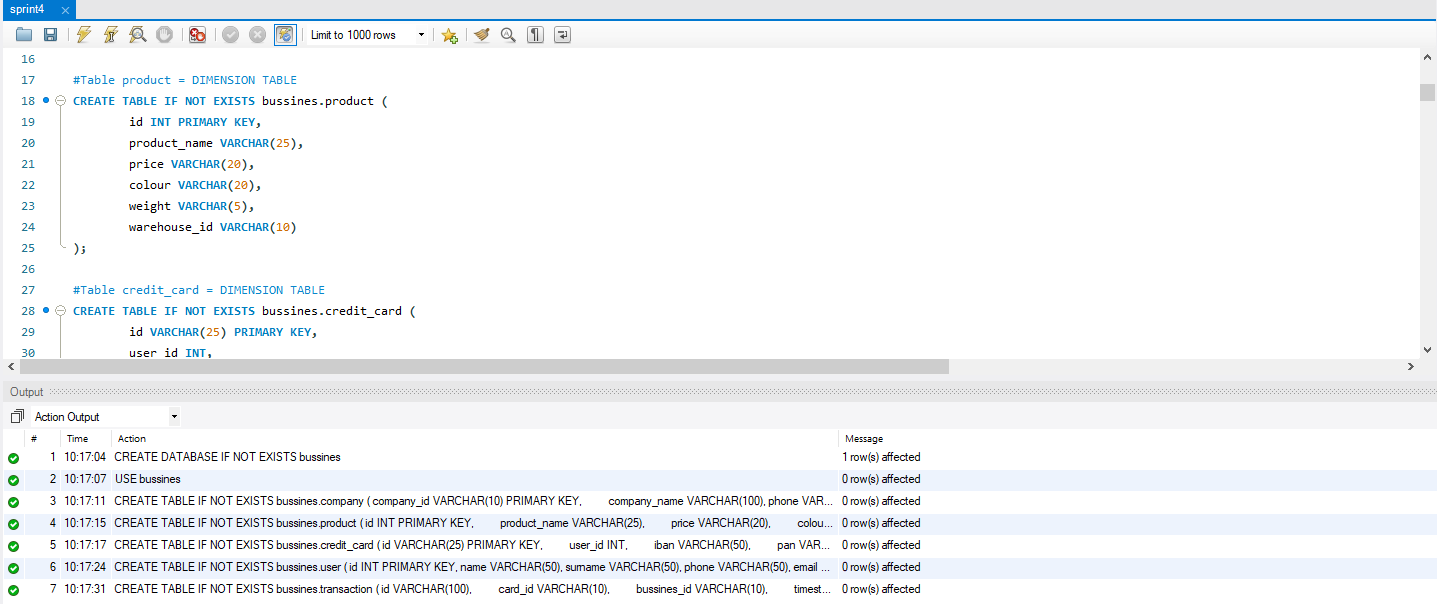
Bussines.company = DIMENSION TABLE

PRIMARY KEY = Company\_id



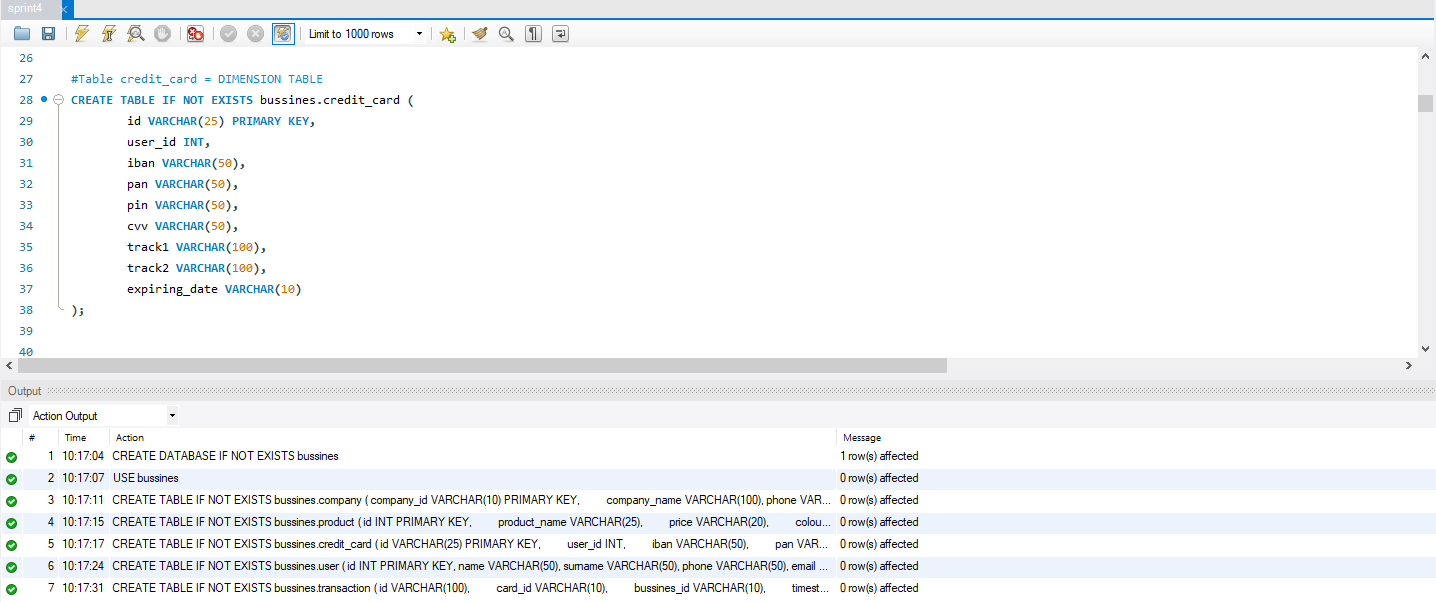
Bussines.product = DIMENSION TABLE

PRIMARY KEY = id



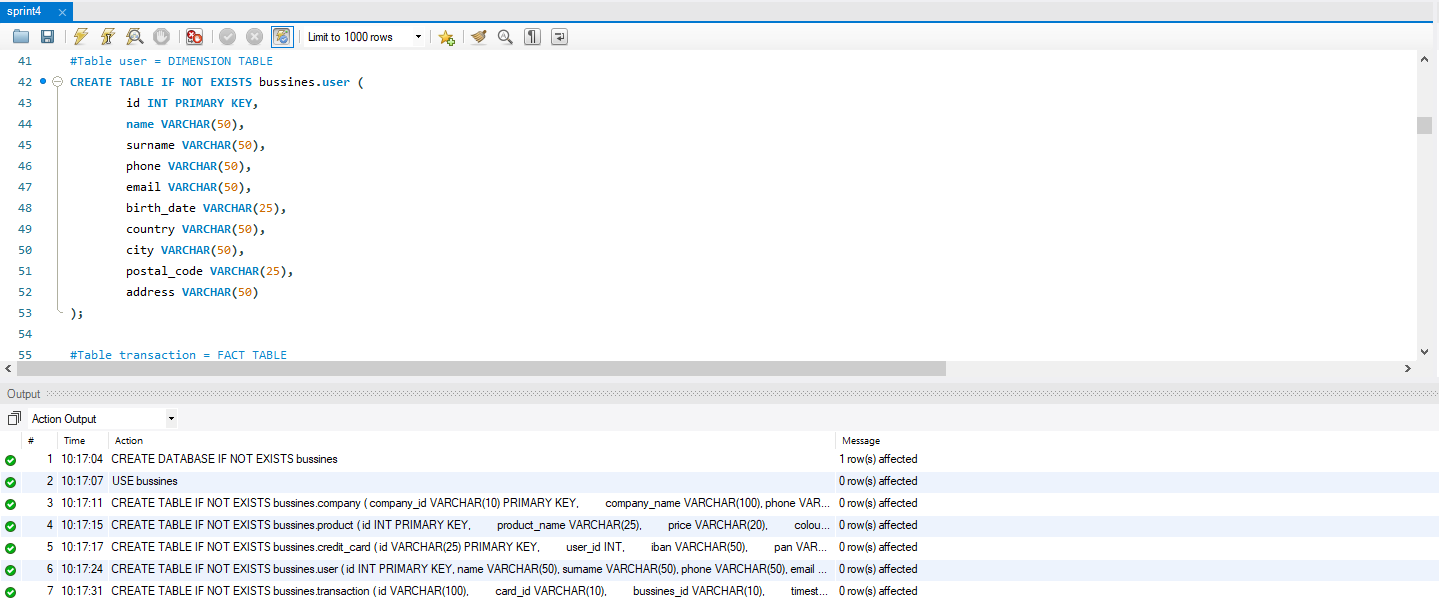
Bussines.credit\_card = DIMENSION TABLE

PRIMARY KEY = id



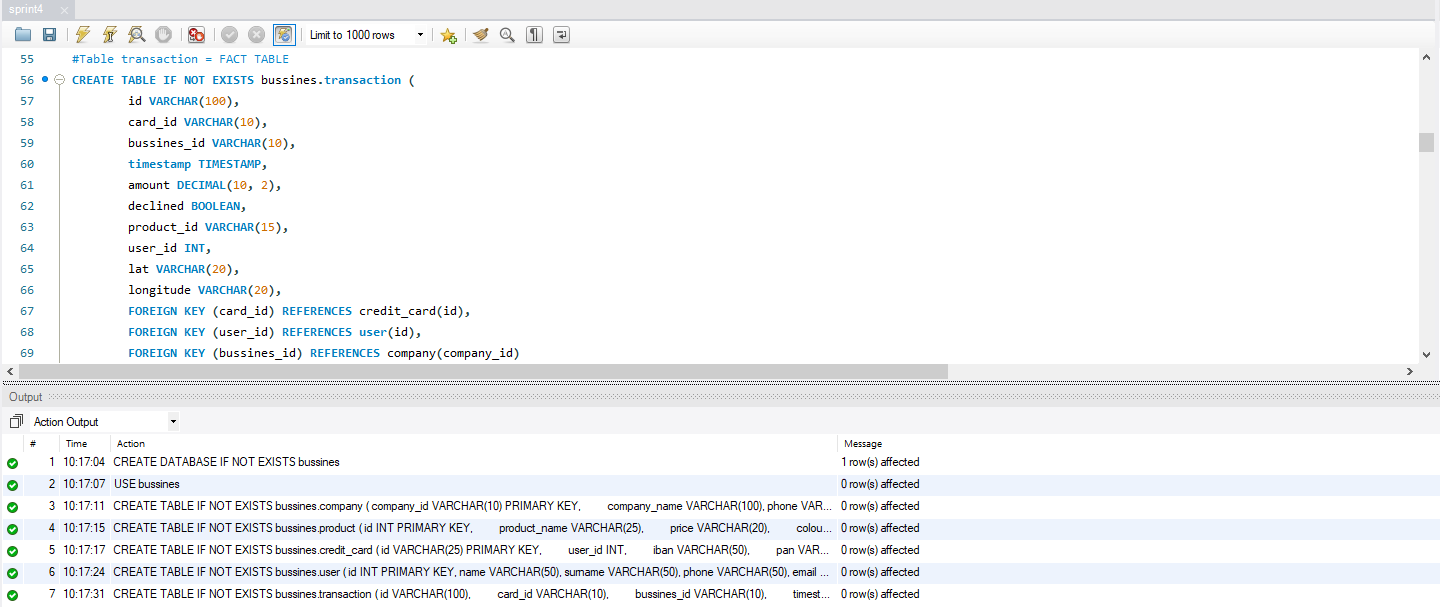
Bussines.user = DIMENSION TABLE

PRIMARY KEY = id



Bussines.transaction = FACT TABLE

PRIMARY KEY = id



En la **FACT TABLE** también estableceremos los **FOREIGN KEY**, es decir las relaciones de esta tabla con sus dimensiones. Serán las siguientes:

1-FOREIGN KEY (card\_id) REFRENCES credit\_card(id) – Relación N to ONE.

2-FOREIGN KEY (user\_id) REFRENCES user(id) – Relación N to ONE.

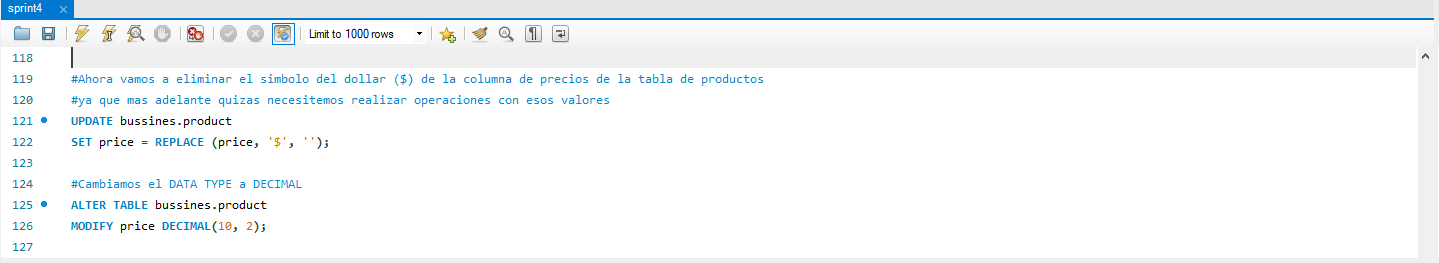
3-FOREIGN KEY (bussines\_id) REFRENCES company(company\_id) – Relación N to ONE.

No estableceremos una relación entre la tabla *product* y la FACT TABLE. Eso es debido a que en la tabla *transaction* la columna *product\_ids* puede contener más de un *id* de producto. Esto establece una relación de MANY to MANY (N to N), la cual vamos a solucionar en el siguiente paso.

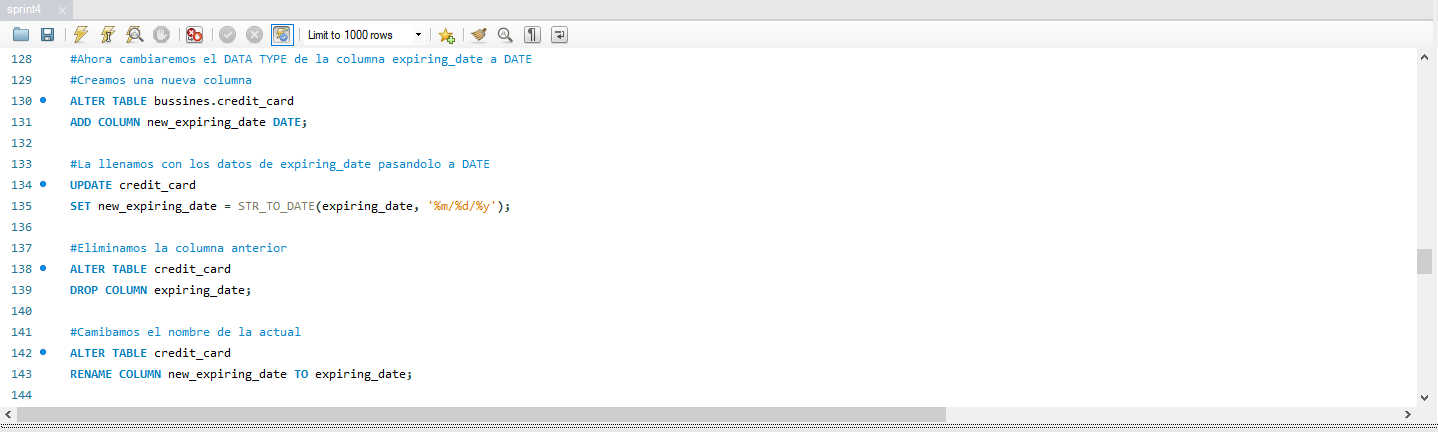
Adecuación de la base de datos

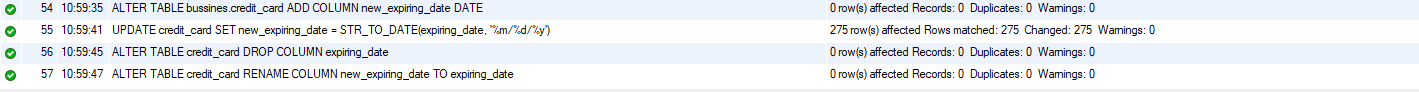
Vamos a realizar varias acciones para limpiar y adecuar los datos de forma que sea mas fácil trabajar con ellos luego.

1.Primero vamos a sacar el símbolo del dollar ($) de la columna de precios de la tabla de productos; de esta forma nos será posible realizar operaciones con la columna en caso de que lo necesitemos.

2.A continuación cambiaremos el **DATA TYPE** de la columna *expiring\_date* en la tabla *credit\_card.* Lo pasaremos a **DATE**.



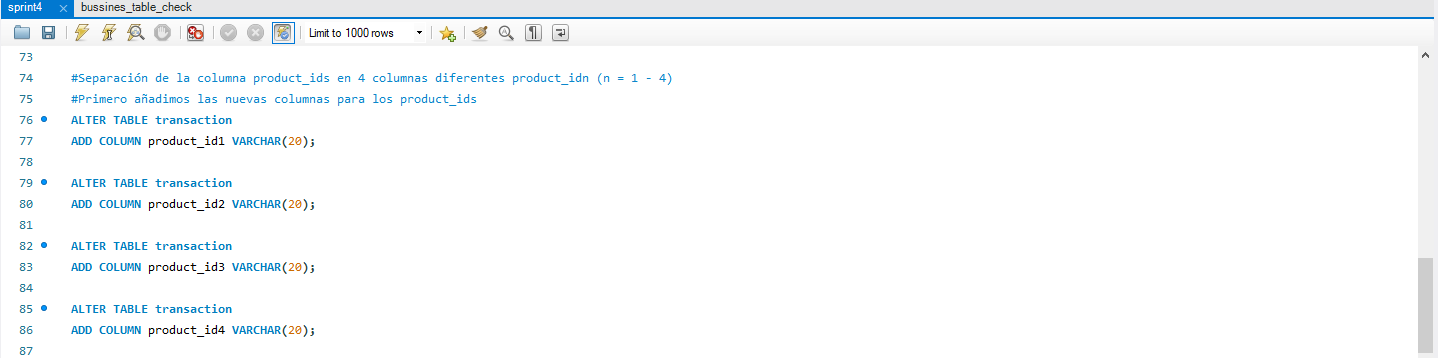


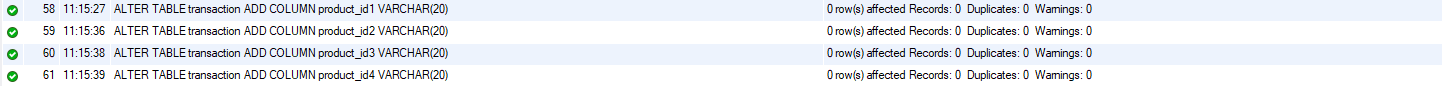
Crearemos una nueva columna en la misma tabla, la llenaremos con los datos de la columna *expiring\_date* usando la expresión **STR\_TO\_DATE**, luego eliminaremos la columna *expiring\_date* y renombraremos la nueva columna creada a *expiring\_date.*

3.Creación de junction table entre transaction y product.

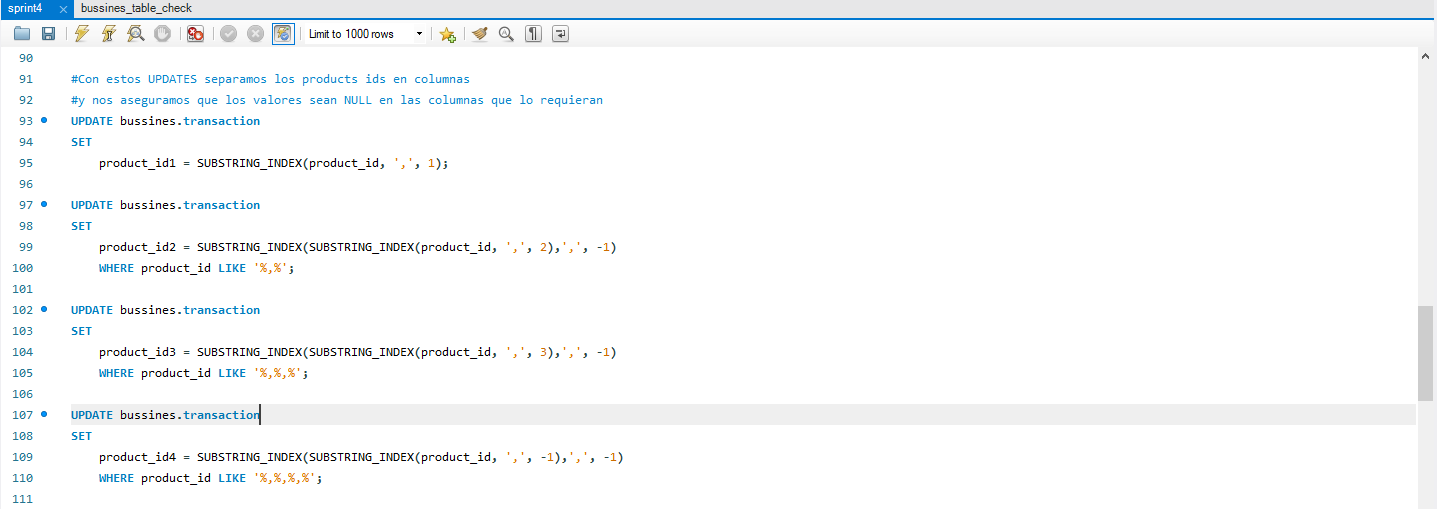
En sistemas de bases de datos relacionales lo más apropiado cuando se encuentran relaciones de tipo **N to N** es crear una tabla intermedia (*junction table*) que permita extraer información de ambas tablas. Esta *junction table* tendrá relaciones N to ONE con la tabla *product* y la tabla *transaction.* Eso nos permitirá simplificar *querys* a la hora de recuperar información y mantener la integridad estructural y lógica de la base de datos, ya que podremos crear FOREIGN KEYS entre las tablas que preservaran la consistencia de la base datos a la hora de introducir nuevos datos en la misma.

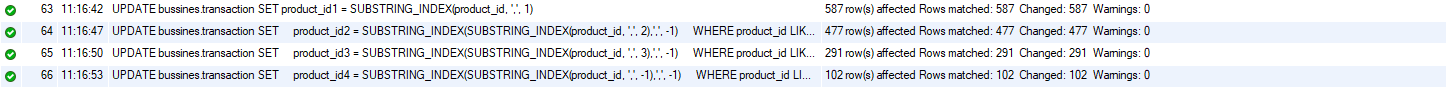
Empezaremos separando la columna *product\_ids* de la tabla *transaction* en 4 columnas diferentes. Para ello crearemos 4 nuevas columnas que podrán tener un valor **INT** o **NULL** en función de si una *transaction* tiene entre 1 o 4 product ids. Estas columnas se llamarán *product\_id1, porudct\_id2, product\_id3, product\_id4.*



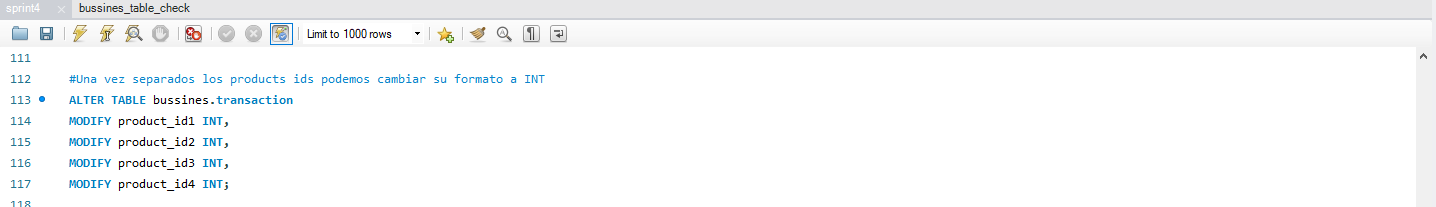


Ahora rellenamos estas columnas con el comando **UPDATE** y la función **SUBSTRING\_INDEX**.





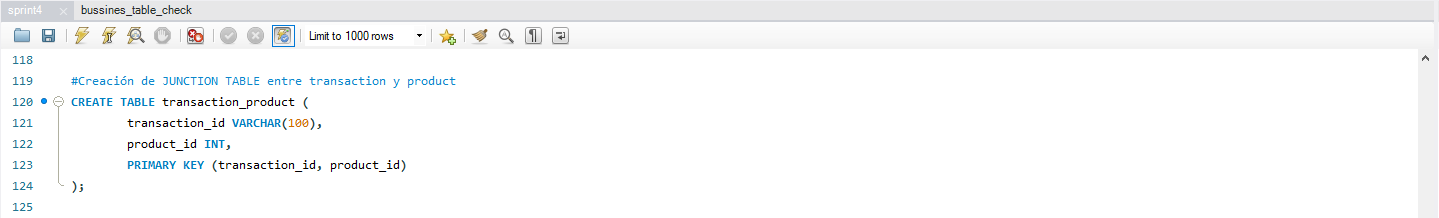
Finalmente definimos la DATA de las nuevas columnas como **INT.**





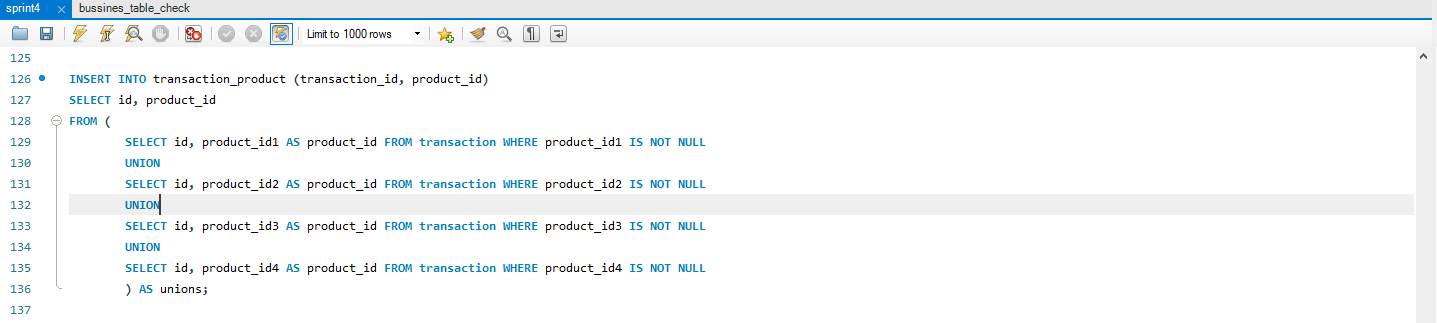
Una vez separados los *product\_id* en diferentes columnas vamos a crear la *junction table* a la cual vamos a llamar *transaction\_product*. Esta tabla va a contener las dos **PRIMARY KEYS** de las tablas *transaction* y *product*; y va a contener cada pareja de *transaction\_id y product\_id* que exista en la tabla *transaction*. Es decir, si la transacción con id 120 contiene las prodcut ids ’15, 25, 35’, nuestra nueva tabla va a contener las *rows* 120-15, 120-25, 120-35; representando así cada pareja existenete entre *transaction\_id y product\_id* en la tabla de *transaction.*

Primero creamos la tabla:



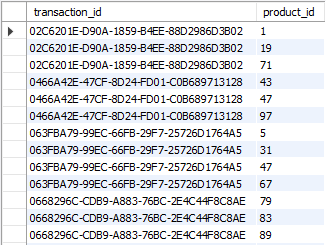


Ahora introducimos los datos en la tabla creada. Como hemos separado las *product\_id* en diferentes columnas este proceso es muy sencillo, con un simple **SELECT** se relacionan las *product\_id* y las *transaction\_id.*



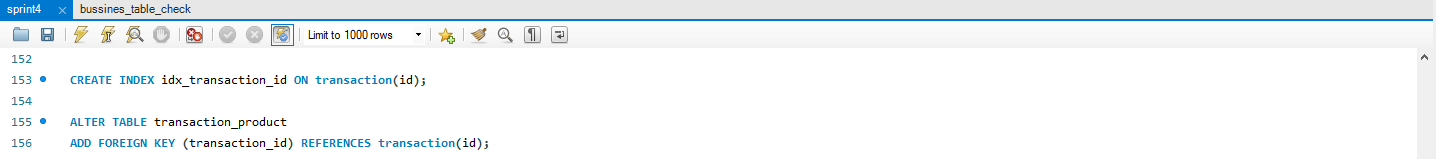


La nueva *junction table transaction\_product* tiene el siguiente formato:

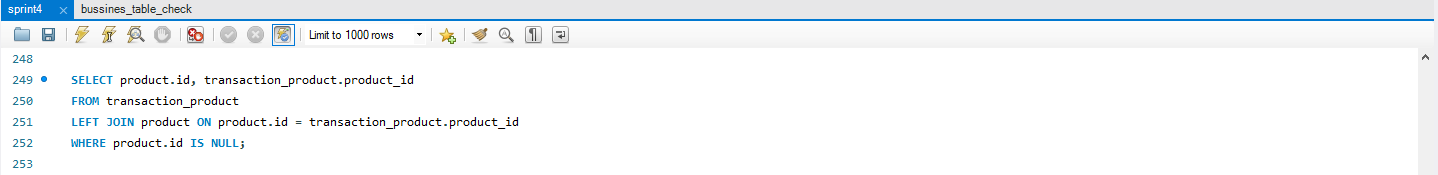


Ahora podemos establecer **FOREIGN KEYS** entre las tablas *transaction, product y transaction\_product.*

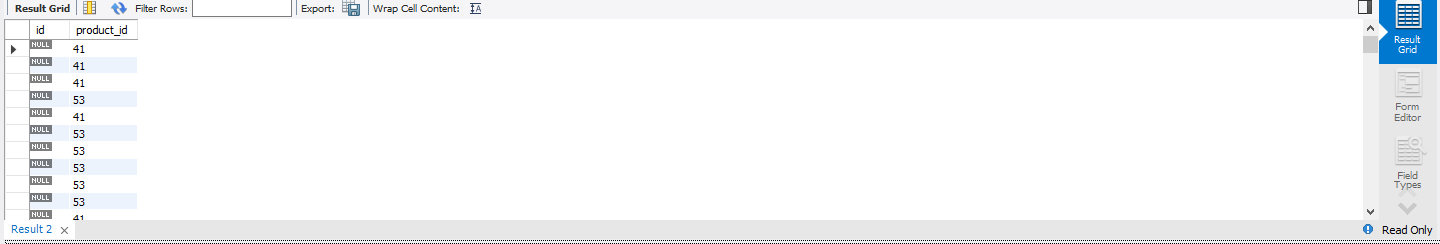
Creamos un índice en la tabla *transaction*, en la columna *id.* A continuación establecemos la relación entre *transaction.id* y *transaction\_product.transaction\_id.*



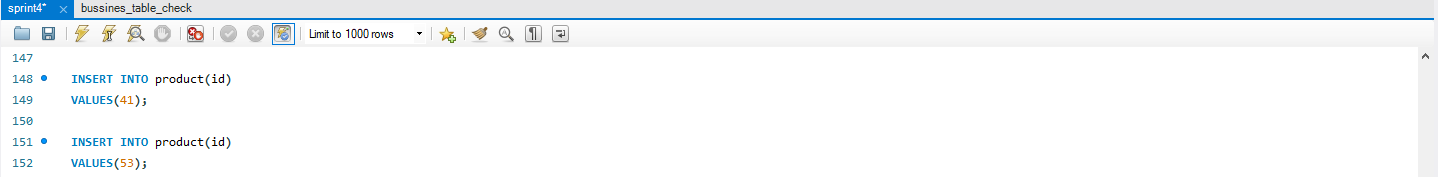
Cuando intentamos establecer la FOREIGN KEY entre las tablas *transaction\_product* y *product*, nos aparece un error que nos indica que la FOREIGN KEY CONSTRAINT no se está respetando. Como estamos intentando establecer la FOREIGN KEY desde la tabla *transaction\_product* esto nos indica que debe haber algunos *product\_id* en la tabla *transaction*, que ahora contan en la tabla *transaction\_product,* que no existen en la tabla *product.* Con una simple query comprobamos cuales son los product\_id que no constan en la tabla *product* y los introducimos en la misma.



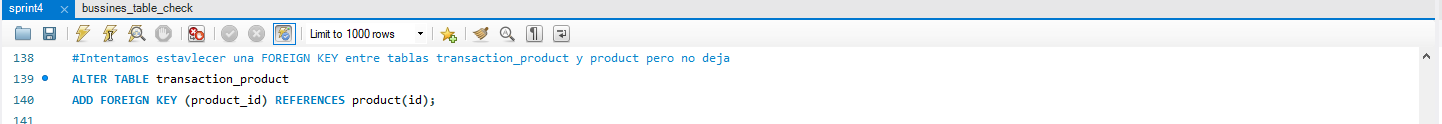




Comprobamos que son las ids 41 y 53. Las introduciermos en la tabla *product*, y así podermos establecer la FOREIGN KEY entre las dos tablas. Si esto fuera un caso real contactaríamos con el departamento de ventas para ver si se ha producido algún error o si falta productos en la base de datos, pero en este caso nos limitaremos a crear los *products* aunque estos no contengan información en la tabla más allá de las ids.



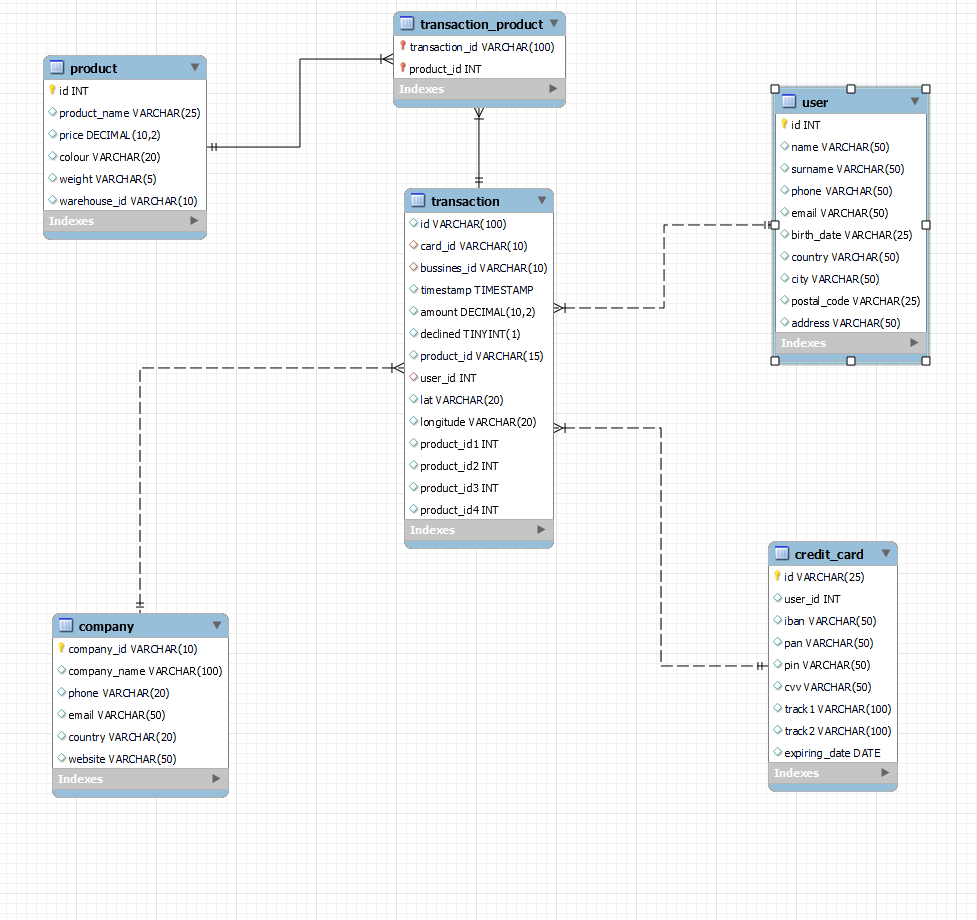






Con este paso final hemos acabado de adecuar nuestra base de datos y esta está lista para ser trabajada.

Este es el esquema final de nuestra base de datos.

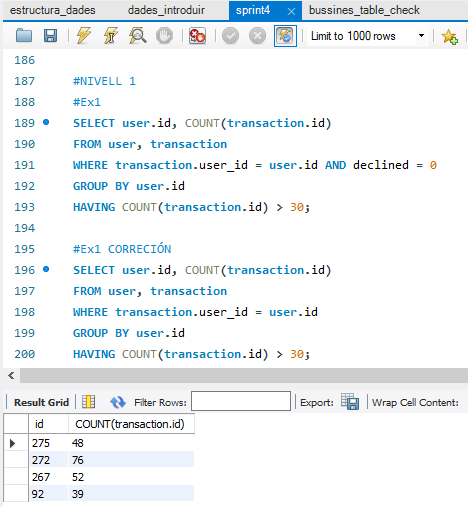


Se trata de un **modelo en estrella** con la tabla *transaction* como FACT TABLE i cuatro DIMENSION TABLES (*user, credit\_card, company, product*) además de una tabla de unión (*junction table*) *transaction\_product* que nos soluciona la relación **N to N** entre las tablas *transaction* y *product*, estableciendo ella misma relaciones **N to ONE** con las tablas *transaction* y *product*.

**NIVELL 1**

**Ex1**

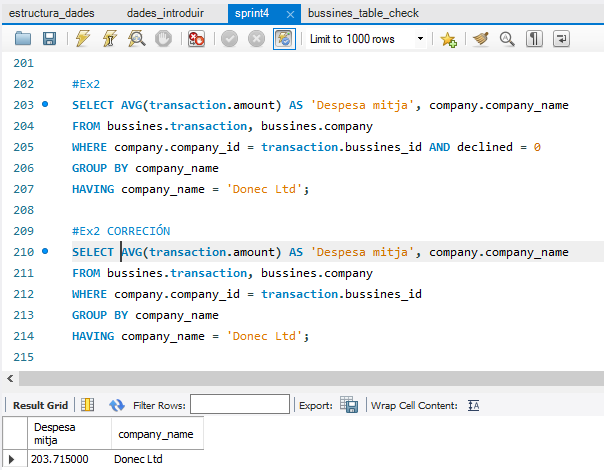
Con un simple **COUNT** podemos y la cláusula **HAVING** para establecer que tenga más de 30 transacciones podemos extraer la información.



\*Se ha añadido la corrección sin la condición de ‘*declined = 0’.*

**Ex2**

Con la fórmula **AVG** y la cláusula **HAVING** para establecer que el nombre de la compañía sea ‘Donec Ltd’ podemos extraer la información.



\*Se ha añadido la corrección sin la condición de ‘*declined = 0’.*

**NIVELL 2**

**Ex1**

Este query va a ser un poco más complejo.

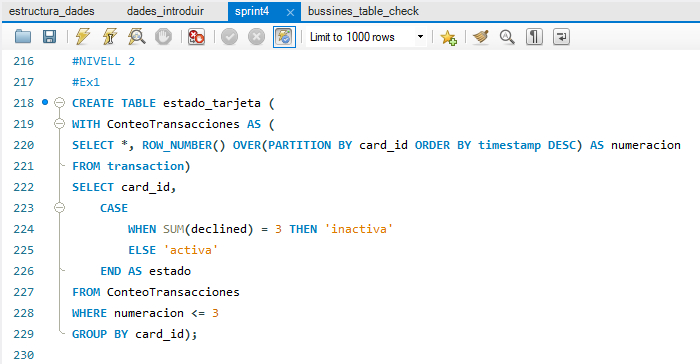
Cuando aparece una columna condicional como en este caso sabemos que probablemente vamos a tener que utilizar el **CASE**. En este caso la condición es que las ultimas 3 transacciones hayan sido declinadas, de este modo sabremos que la tarjeta esta inactiva.

Para extraer las últimas 3 transacciones vamos a tener que numerar las *rows* de la tabla *transaction.* Para ello vamos a usar la cláusula **ROW\_NUMBER()**, y esta numeración la vamos a hacer sobre (**OVER**)la columna *card\_id,* ordenándola por *timestamp*, para que nos la ordene temporalmente.

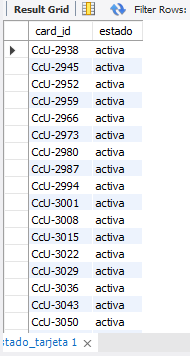
La query finalmente va a ser de la siguiente forma:

Primero utilizaremos **ROW\_NUMBER** para numerar y oredenar temporalmente por *card\_id* las trasnacciones. A esto lo llamaremos *numeració,* y servirá para establecer nuestra condición (últimas 3 transacciones). Con esto haremos un **SELECT** de las *card\_id* utilizando un **CASE** donde la condición será que la **SUM**(*declined*) sea menor que 3 (ya que para que una tarjeta este inactiva esta tiene que tener las últimas tres transacciones declinadas). Finalmente, con el **WHERE** estableceremos que *numeració <= 3,* para quedarnos solo con las 3 últimas transacciones.

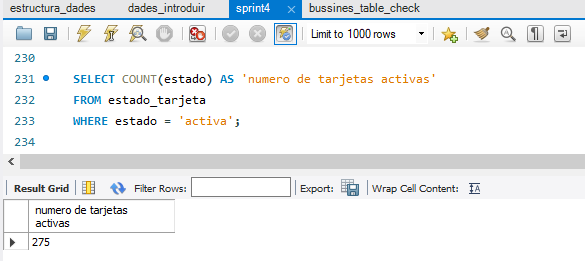
Todo esto irá dentro de un **CREATE TABLE** para que nos lo transforme todo en una tabla, a la cual llamaremos *estado\_tarjeta.*



La tabla resultante tiene este formato:



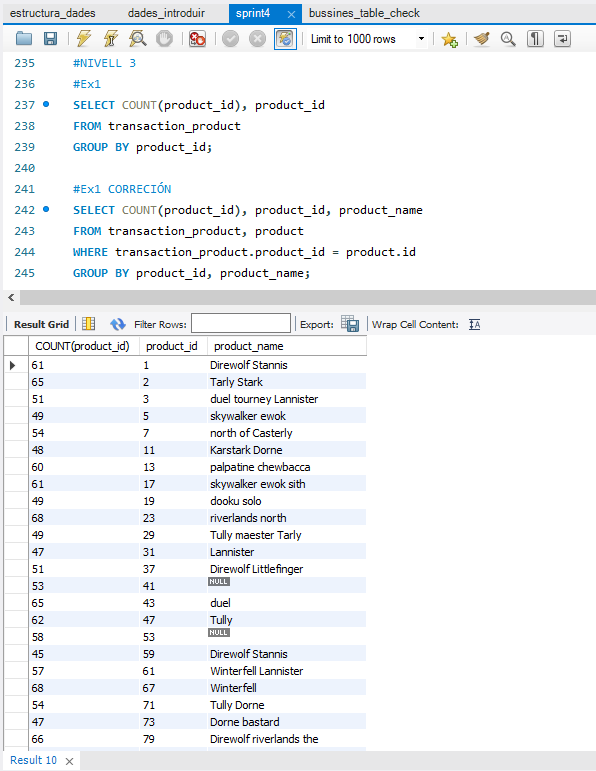
Con un **COUNT** podemos saber cuántas tarjetas activas tenemos.



**NIVELL 3**

**Ex1**

Gracias a la *junction table* que hemos creado antes (*transaction\_product*) la query necesaria para saber cuántos productos hemos vendido es un simple **COUNT** agrupado por *product\_id*.



\*Se ha añadido una corrección para mostrar el nombre del producto en la misma *query*. Los productos con *id* 41 y 53 no tienen nombre porque son los que hemos añadido nosotros para poder establecer las *FOREIGN KEYS* entre las tablas *transaction\_product* y *product*, ya que los registros 41 y 53 existian en la tabla *transaction*pero no en la *product*.