

Tasca S4.01. Creació de Base de Dades

•Descripció

•Partint d'alguns arxius CSV dissenyaràs i crearàs la teva base de dades.

•Important

•Totes les transformacions i importacions que se't demanen en aquesta tasca s'han de realitzar **utilitzant codi SQL. NO ES PERMET fer els canvis fent servir el Wizard.**

•Nivell 1

•Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules de les quals puguis realitzar les següents consultes:

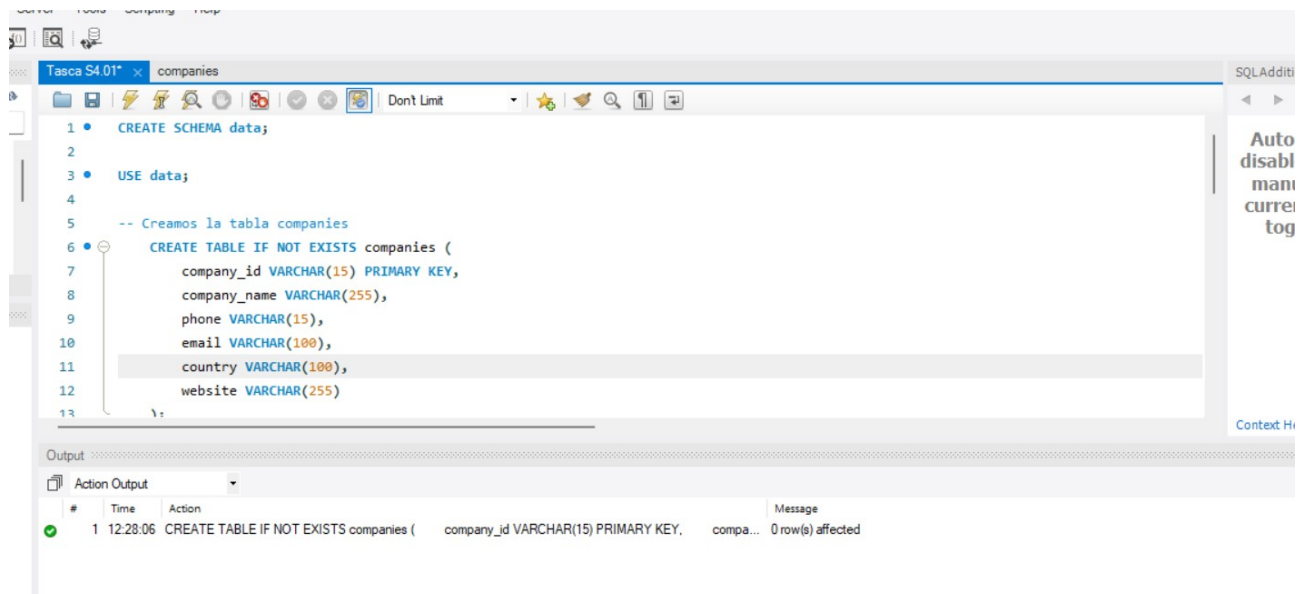
- Primero de todo creamos una nueva base de datos con el siguiente código:

```
CREATE SCHEMA data;
```

```
USE data;
```

- Empezamos a crear las tablas revisando los distintos CSV.

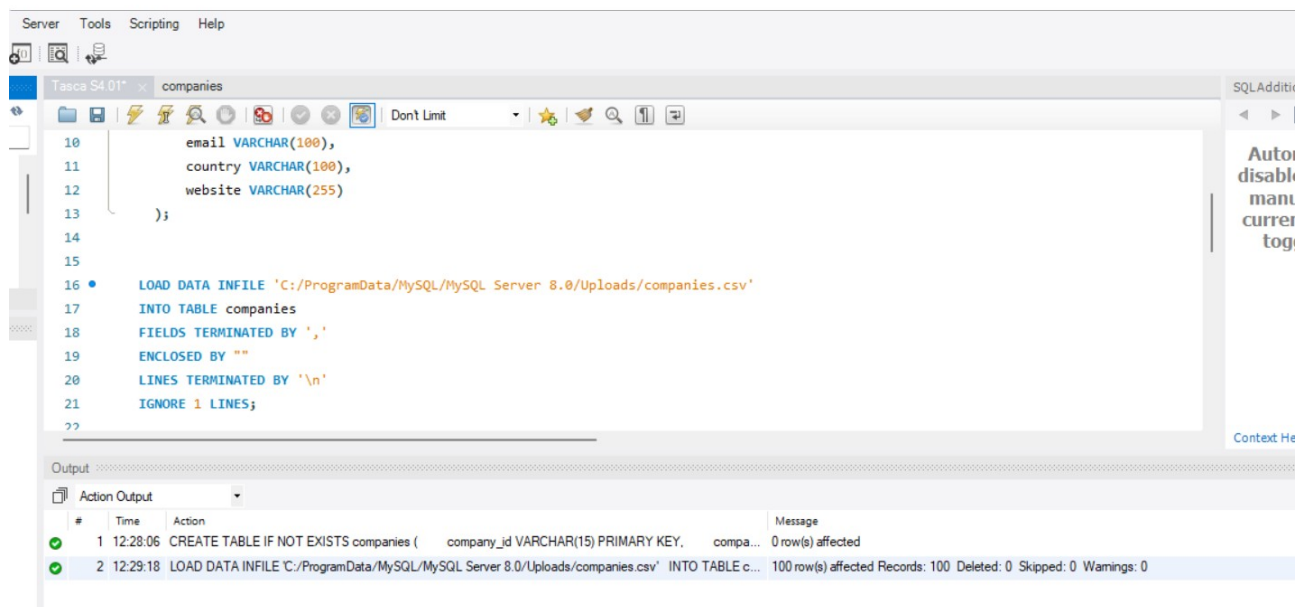
- Creamos la tabla companies



- Previo a cargar los datos usamos el siguiente código

SHOW VARIABLES LIKE "secure_file_priv"; Este código nos da la ruta del directorio habilitado para hacer importaciones y exportaciones.

- Cargamos los datos de la tabla companies



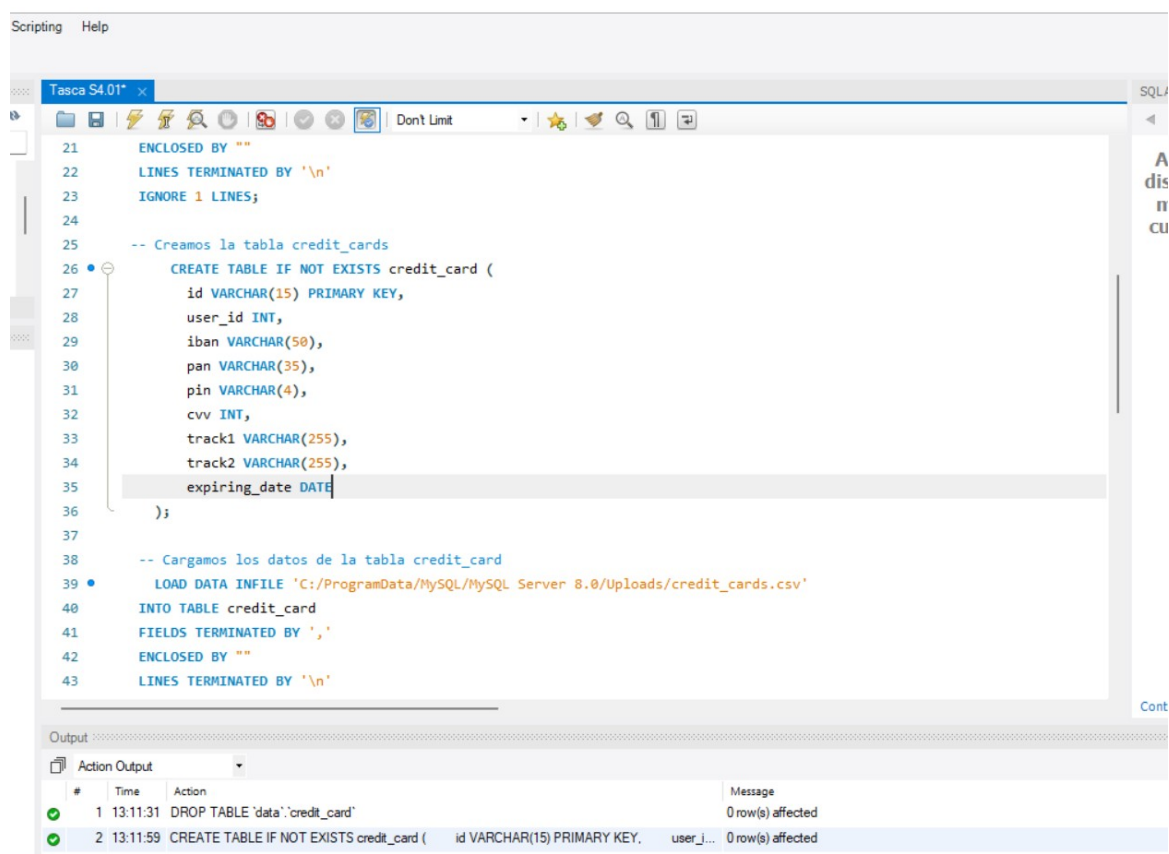
The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a script for creating the 'companies' table and loading data from a CSV file. The bottom pane shows the execution output.

```
10      email VARCHAR(100),
11      country VARCHAR(100),
12      website VARCHAR(255)
13  );
14
15
16  • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'
17  INTO TABLE companies
18  FIELDS TERMINATED BY ','
19  ENCLOSED BY '"'
20  LINES TERMINATED BY '\n'
21  IGNORE 1 LINES;
```

Output:

#	Time	Action	Message
1	12:28:06	CREATE TABLE IF NOT EXISTS companies (company_id VARCHAR(15) PRIMARY KEY, compa...	0 row(s) affected
2	12:29:18	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv' INTO TABLE c...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

- Creamos la tabla credit_cards



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a script for creating the 'credit_cards' table and loading data from a CSV file. The bottom pane shows the execution output.

```
21  ENCLOSED BY '"'
22  LINES TERMINATED BY '\n'
23  IGNORE 1 LINES;
24
25  -- Creamos la tabla credit_cards
26  • CREATE TABLE IF NOT EXISTS credit_card (
27      id VARCHAR(15) PRIMARY KEY,
28      user_id INT,
29      iban VARCHAR(50),
30      pan VARCHAR(35),
31      pin VARCHAR(4),
32      cvv INT,
33      track1 VARCHAR(255),
34      track2 VARCHAR(255),
35      expiring_date DATE
36  );
37
38  -- Cargamos los datos de la tabla credit_card
39  • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
40  INTO TABLE credit_card
41  FIELDS TERMINATED BY ','
42  ENCLOSED BY '"'
43  LINES TERMINATED BY '\n'
```

Output:

#	Time	Action	Message
1	13:11:31	DROP TABLE 'data'.credit_card	0 row(s) affected
2	13:11:59	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, user_j...	0 row(s) affected

- Cargamos los datos de la tabla credit_cards y realizamos el cambio de formato de fecha de expiring_date al formato correcto de Mysql.

The screenshot shows a SQL script in a text editor. The script includes comments in Spanish and SQL commands to load data from a CSV file into a table named 'credit_card' and to create a table named 'users'. The 'credit_card' table has columns: id, user_id, iban, pan, pin, cvv, track1, track2, and expiring_date. The 'users' table has columns: id, name, surname, phone, email, birth_date, and address. The output window shows the execution results of these commands.

```

33     track1 VARCHAR(255),
34     track2 VARCHAR(255),
35     expiring_date DATE
36 );
37
38 -- Cargamos los datos de la tabla credit_card
39 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
40 INTO TABLE credit_card
41 FIELDS TERMINATED BY ','
42 ENCLOSED BY '"'
43 LINES TERMINATED BY '\n'
44 IGNORE 1 ROWS
45 (id, user_id, iban, pan, pin, cvv, track1, track2, @expiring_date)
46 SET expiring_date = STR_TO_DATE(@expiring_date, '%m/%d/%y');
47
48 -- Creamos la tabla users
49 • CREATE TABLE IF NOT EXISTS users (
50     id INT PRIMARY KEY,
51     name VARCHAR(100),
52     surname VARCHAR(100),
53     phone VARCHAR(150),
54     email VARCHAR(150),
55     birth_date VARCHAR(100),
56     address VARCHAR(255)
57 );

```

Output:

#	Time	Action	Message
1	13:11:31	DROP TABLE 'data'.'credit_card'	0 row(s) affected
2	13:11:59	CREATE TABLE IF NOT EXISTS credit_card (id VARCHAR(15) PRIMARY KEY, user_id VARCHAR(15) PRIMARY KEY, iban VARCHAR(15), pan VARCHAR(15), pin VARCHAR(15), cvv VARCHAR(15), track1 VARCHAR(255), track2 VARCHAR(255), expiring_date DATE)	0 row(s) affected
3	13:18:56	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv' INTO TABLE credit_card	5000 row(s) affected Records: 5000 Deleted: 0 Skipped: 0 Warnings: 0

- Creamos la tabla users

The screenshot shows a SQL script in a text editor. The script includes a comment in Spanish and a SQL command to create a table named 'users'. The 'users' table has columns: id, name, surname, phone, email, birth_date, city, country, postal_code, and address. The output window shows the execution result of this command.

```

45 -- Creamos la tabla users
46 • CREATE TABLE IF NOT EXISTS users (
47     id INT PRIMARY KEY,
48     name VARCHAR(100),
49     surname VARCHAR(100),
50     phone VARCHAR(150),
51     email VARCHAR(150),
52     birth_date VARCHAR(100),
53     city VARCHAR(150),
54     country VARCHAR(150),
55     postal_code VARCHAR(100),
56     address VARCHAR(255)
57 );

```

Output:

#	Time	Action	Message
1	12:48:13	CREATE TABLE IF NOT EXISTS users (id INT PRIMARY KEY, name VARCHAR(100), surname VARCHAR(100), phone VARCHAR(150), email VARCHAR(150), birth_date VARCHAR(100), city VARCHAR(150), country VARCHAR(150), postal_code VARCHAR(100), address VARCHAR(255))	0 row(s) affected

- Cargamos los datos en la tabla users de los 2 csv, american_users y european_users
- Primero cargamos los datos CSV de european_users y al final del código usamos el

SET birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y') para transformar la cadena que esta en birth_date al formato de fecha MySql.

Donde %b quiere decir mes abreviado, %d dia del mes en 2 digitos y %Y año con 4 dígitos.

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL script with the following content:

```
-- Cargamos los datos de las tablas european_users y american_users
LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
IGNORE 1 ROWS
(id, name, surname, phone, email, @birth_date, country, city, postal_code, address)
SET birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y');
```

The bottom pane shows the 'Result Grid' with the following data:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	1980-07-02	United Kingdom	London	EC1A 1BB	Ap #432-4493 /
152	Hakeem	Alford	(0111) 367 0184	adpiscng.ligula@google.edu	1979-09-30	United Kingdom	Birmingham	B1 1AA	551-8930 Labor
153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	1994-07-27	United Kingdom	London	EC1A 1BB	Ap #312-5898 C
154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	1986-11-02	United Kingdom	Manchester	M1 1AE	872-1866 Pede l
155	Joshua	Russell	055 4409 5286	justo.nec.ante@outlook.edu	1984-01-23	United Kingdom	Manchester	M1 1AE	Ap #285-4727 /
156	Remedios	Case	055 3114 1566	mollis.phasellus.libero@aol.com	1994-10-09	United Kingdom	Liverpool	L1 1AA	479-3690 Turpis
157	Philip	Carey	0800 640 6251	phasellus@yahoo.net	1992-10-10	United Kingdom	Manchester	M1 1AE	196-1103 Quisq

The 'Output' pane shows the following messages:

```
# Time Action Message Duration
1 12:30:41 CREATE TABLE IF NOT EXISTS users (id INT PRIMARY KEY, name VARCHAR(100), surname... 0 row(s) affected 0.031 sec
2 12:32:36 LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_users.csv'... Error Code: 1054. Unknown column 'user_name' in 'field list' 0.015 sec
3 12:33:28 LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_users.csv'... Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot exec... 0.015 sec
4 12:34:24 LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_user... 3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0 0.141 sec
5 12:35:53 SELECT * FROM users 3990 row(s) returned 0.000 sec
```

- Realizamos la misma acción con el CSV de american_users

- Y comprobamos que se hayan cargado bien los datos.

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL script with the following content:

```
LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
IGNORE 1 ROWS
(id, name, surname, phone, email, @birth_date, country, city, postal_code, address)
SET birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y');
```

The bottom pane shows the 'Result Grid' with the following data:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	1985-11-17	United States	New York	10001	348-7818 Sagitts t
2	Garrett	Mconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	1992-08-23	United States	Philadelphia	19101	903 Sit Ave
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	1998-04-29	United States	Houston	77001	736-2063 Tellus St
4	Howard	Stafford	1-411-740-3269	omare.egestas@icloud.edu	1989-02-18	United States	Phoenix	85001	Ap #545-2244 Era
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	1998-09-26	United States	Philadelphia	19101	341-2821 Ultrices /
6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	1989-10-15	United States	San Jose	95101	888-2799 Amet Str
7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	1981-12-04	United States	Chicago	60601	8627 Malesuada Ri
8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	1993-08-01	United States	New York	10001	Ap #251-7144 Inte
9	Mannix	Mclain	(590) 883-2184	aliquam.nisi@outlook.com	1987-01-24	United States	San Antonio	78201	647-3080 Lacus. S

The 'Output' pane shows the following messages:

```
# Time Action Message Duration
1 12:48:29 LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_users... 1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0
2 12:48:56 SELECT * FROM users 5000 row(s) returned
```

- Realizamos el cambio de tipo de dato en la columna birth_date de Varchar a Date, y comprobamos.

SQL Server Enterprise Manager - Script Editor

```

74 IGNORE 1 ROWS
75 (id, name, surname, phone, email, @birth_date, country, city, postal_code, address)
76 SET birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y');
77
78 -- Realizamos el cambio de dato de la columna birth_date de varchar como lo habiamos creado a DATE.
79
80 ALTER TABLE users
81 MODIFY COLUMN birth_date DATE;
82 DESCRIBE users;
83
84

```

Result Grid

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI		
name	varchar(100)	YES			
surname	varchar(100)	YES			
phone	varchar(150)	YES			
email	varchar(150)	YES			
birth_date	date	YES			
country	varchar(150)	YES			
city	varchar(150)	YES			
postal_code	varchar(100)	YES			
address	varchar(255)	YES			

Output

#	Time	Action	Message	Duration
1	12:48:29	LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\american_use...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0	0.078 s
2	12:48:56	SELECT * FROM users	5000 row(s) returned	0.015 s
3	12:50:32	DESCRIBE users	10 row(s) returned	0.000 s
4	12:55:36	ALTER TABLE users MODIFY COLUMN birth_date DATE	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0	0.156 s

- Creamos la tabla Transactions

SQL Server Enterprise Manager - Script Editor

```

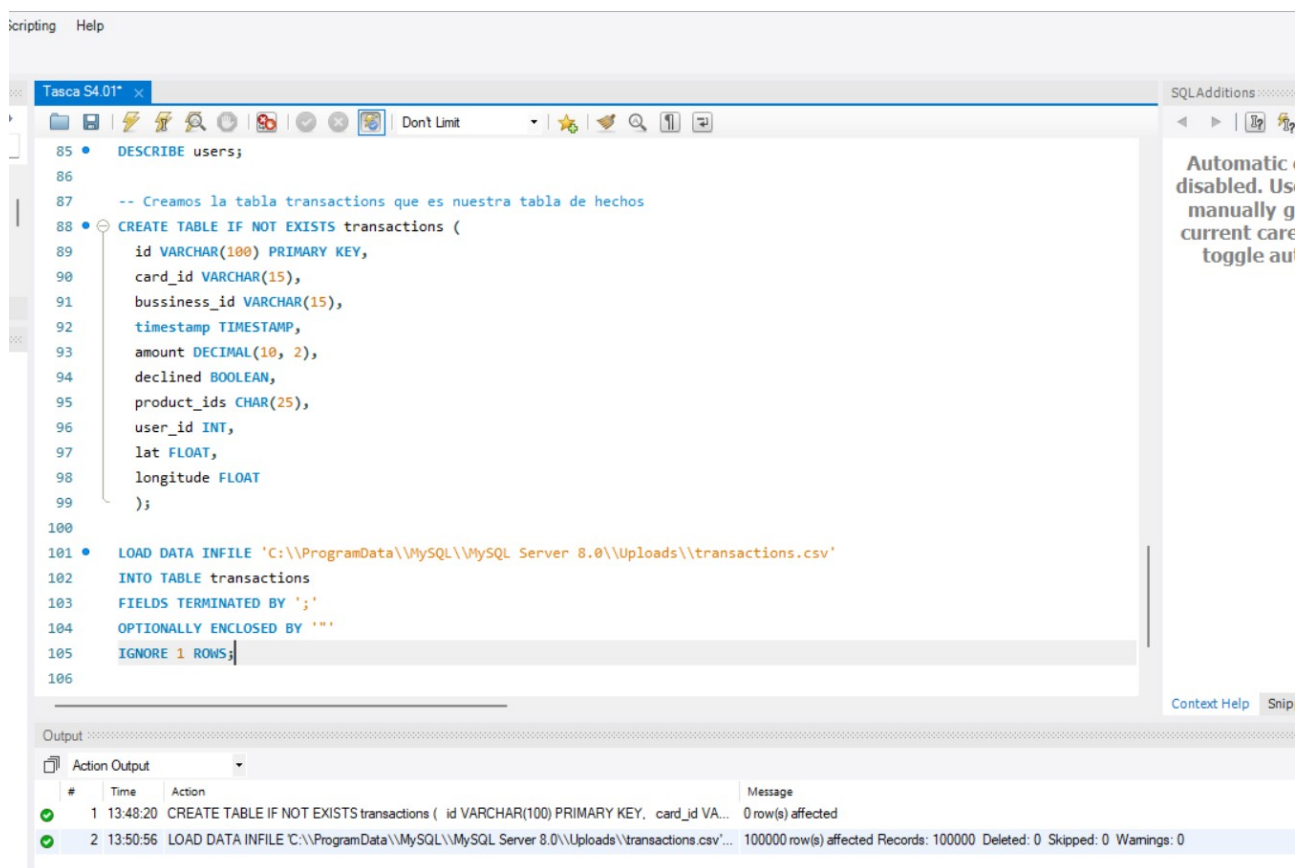
85 DESCRIBE users;
86
87 -- Creamos la tabla transactions que es nuestra tabla de hechos
88 CREATE TABLE IF NOT EXISTS transactions (
89     id VARCHAR(100) PRIMARY KEY,
90     card_id VARCHAR(15),
91     bussiness_id VARCHAR(15),
92     timestamp TIMESTAMP,
93     amount DECIMAL(10, 2),
94     declined BOOLEAN,
95     product_ids CHAR(25),
96     user_id INT,
97     lat FLOAT,
98     longitude FLOAT
99 );
100
101
102
103
104
105
106

```

Output

#	Time	Action	Message
1	13:48:20	CREATE TABLE IF NOT EXISTS transactions (id VARCHAR(100) PRIMARY KEY, card_id VA...	0 row(s) affected

- Cargamos los datos de la tabla transactions



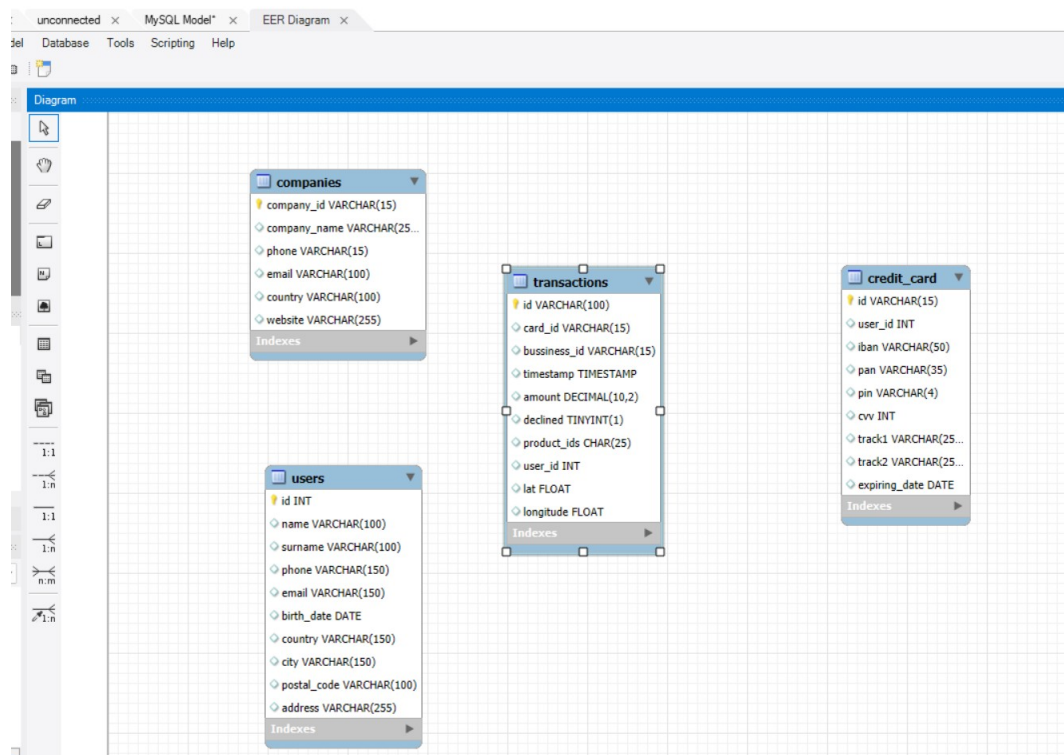
The screenshot shows a SQL script editor with a toolbar at the top. The script contains the following SQL commands:

```
85 • DESCRIBE users;
86
87 -- Creamos la tabla transactions que es nuestra tabla de hechos
88 • CREATE TABLE IF NOT EXISTS transactions (
89     id VARCHAR(100) PRIMARY KEY,
90     card_id VARCHAR(15),
91     bussiness_id VARCHAR(15),
92     timestamp TIMESTAMP,
93     amount DECIMAL(10, 2),
94     declined BOOLEAN,
95     product_ids CHAR(25),
96     user_id INT,
97     lat FLOAT,
98     longitude FLOAT
99 );
100
101 • LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv'
102 INTO TABLE transactions
103 FIELDS TERMINATED BY ','
104 OPTIONALLY ENCLOSED BY '"'
105 IGNORE 1 ROWS;
```

Below the script, the 'Output' pane shows the execution results:

#	Time	Action	Message
1	13:48:20	CREATE TABLE IF NOT EXISTS transactions (id VARCHAR(100) PRIMARY KEY, card_id VA...	0 row(s) affected
2	13:50:56	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv'...	100000 row(s) affected Records: 100000 Deleted: 0 Skipped: 0 Warnings: 0

- Visualizamos el EER a ver como están las tablas



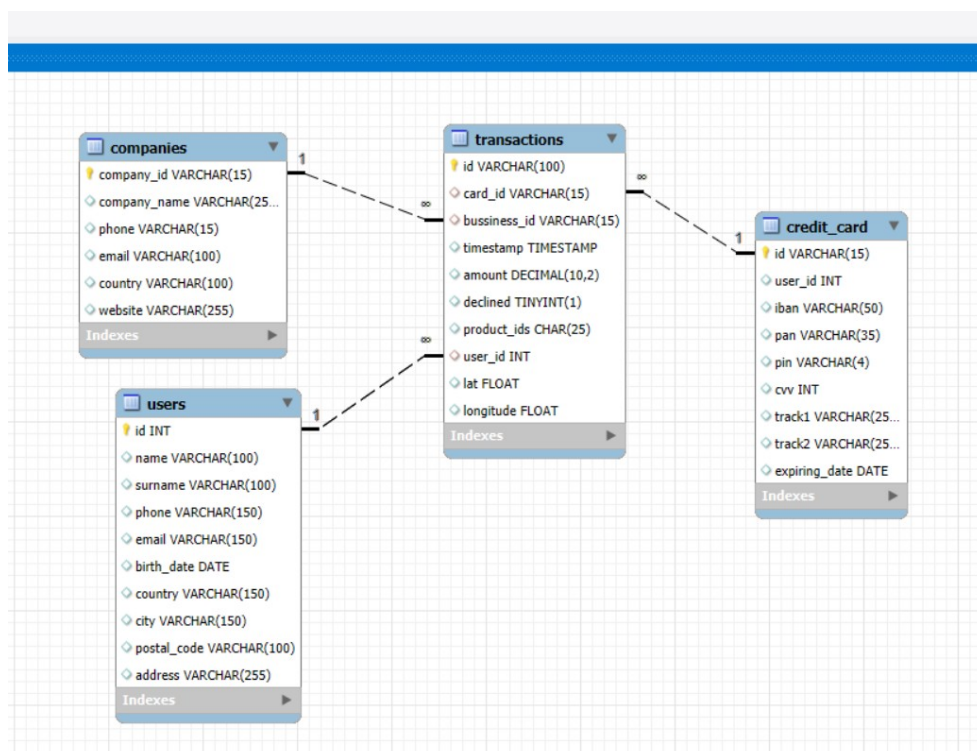
- Ahora realizaremos las conexiones entre las tablas para que nos quede un modelado de estrella tal y como nos pide el ejercicio.

```
102 INTO TABLE transactions
103 FIELDS TERMINATED BY ','
104 OPTIONALLY ENCLOSED BY '"'
105 IGNORE 1 ROWS;
106 -- Realizamos la conexiones entre la tabla de hechos transaction y las tablas de dimensiones restantes
107
108 • ALTER TABLE transactions
109 ADD CONSTRAINT fk_transactions_users
110 FOREIGN KEY (user_id)
111 REFERENCES users(id);
112
113 • ALTER TABLE transactions
114 ADD CONSTRAINT fk_transactions_credit_card
115 FOREIGN KEY (card_id)
116 REFERENCES credit_card(id);
117
118 • ALTER TABLE transactions
119 ADD CONSTRAINT fk_transactions_companies
120 FOREIGN KEY (bussiness_id)
121 REFERENCES companies(company_id);
122
123
```

Output

#	Time	Action	Message
2	14:06:02	ALTER TABLE transactions ADD CONSTRAINT fk_transactions_users FOREIGN KEY (user_id)...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
3	14:07:50	ALTER TABLE transactions ADD CONSTRAINT fk_transactions_credit_card FOREIGN KEY (c...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
4	14:09:53	ALTER TABLE transactions ADD CONSTRAINT fk_transactions_companies FOREIGN KEY (bu...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

- Volvemos a chequear el diagrama EER para visualizar el esquema final y podemos observar el modelado de estrella como se nos había pedido. Con las relaciones de companies a transactions de 1 a N, de users a transactions de 1 a N y de credit_card a transaction de 1 a N.



· Exercici 1

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

- Para este ejercicio hemos realizado la subconsulta en el JOIN con el objetivo de obtener una nueva tabla en la que solicitamos todos los user_id de la tabla transactions con mas de 80 transactions realizadas y aceptadas

SUBCONSULTA :

(SELECT t.user_id, COUNT(t.id) AS total_transactions

FROM transactions t

WHERE declined = 0

GROUP BY t.user_id

HAVING total_transactions > 80

) AS ts -- Transaction superior a 80

- En la consulta principal realizamos un SELECT CONCAT para obtener en una sola columna el nombre completo y el total_transactions para ver que sean mayores a 80 de la tabla users y realizamos el JOIN con la subconsulta (tabla temporal ts) para obtener solo los resultados que cumplen las condiciones de la subconsulta.

The screenshot shows a SQL IDE window titled 'Tasca S4.01*'. The query editor contains the following SQL code:

```
123 -- Exercici 1
124 -- Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.
125
126 • SELECT CONCAT(u.name, ' ', u.surname) AS nombre_completo, total_transactions
127 FROM users u
128 JOIN
129 (
130     SELECT t.user_id, COUNT(t.id) AS total_transactions
131     FROM transactions t
132     WHERE declined = 0
133     GROUP BY t.user_id
134     HAVING total_transactions > 80
135 ) AS ts -- transaction superior a 80
136 ON u.id = ts.user_id;
```

The 'Result Grid' shows the following data:

nombre_completo	total_transactions
Molly Gilliam	107
Dxxwgi Hwcrw	91
Bnyr Astuw	86

The 'Output' pane shows the execution message: '1 11:31:26 SELECT CONCAT(u.name, ' ', u.surname) AS nombre_completo, total_transactions FROM users u ... 3 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.

- Para la realización de esta consulta tenemos que realizar en el SELECT solicitamos que nos devuelva el iban, el nombre de la compañía y la media del amount mediante el ROUND(AVG) para que en este caso solo nos devuelva 2 decimales. Seguidamente realizamos los joins entre las tablas transactions, companies y credit_card poniendo en el WHERE las condiciones para que pertenezcan a la compañía 'Donec Ltd' y que la transaccion haya sido aceptada. Realizamos la agrupación por el IBAN de la tabla credit_card y finalmente ordenamos mediante el ORDER BY la media_amount en orden descendente para que las que tienen mayor amount aparezcan las primeras.

The screenshot shows a SQL IDE window titled 'Tasca S4.01'. The query editor contains the following SQL code:

```
140
141
142 -- Exercici 2
143 -- Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.
144
145 • SELECT cc.iban, c.company_name, ROUND(AVG(t.amount), 2) AS media_amount
146 FROM transactions t
147 INNER JOIN credit_card cc ON t.card_id = cc.id
148 INNER JOIN companies c ON t.business_id = c.company_id
149 WHERE c.company_name = 'Donec Ltd' AND t.declined = 0
150 GROUP BY cc.iban
151 ORDER BY media_amount DESC;
152
153
```

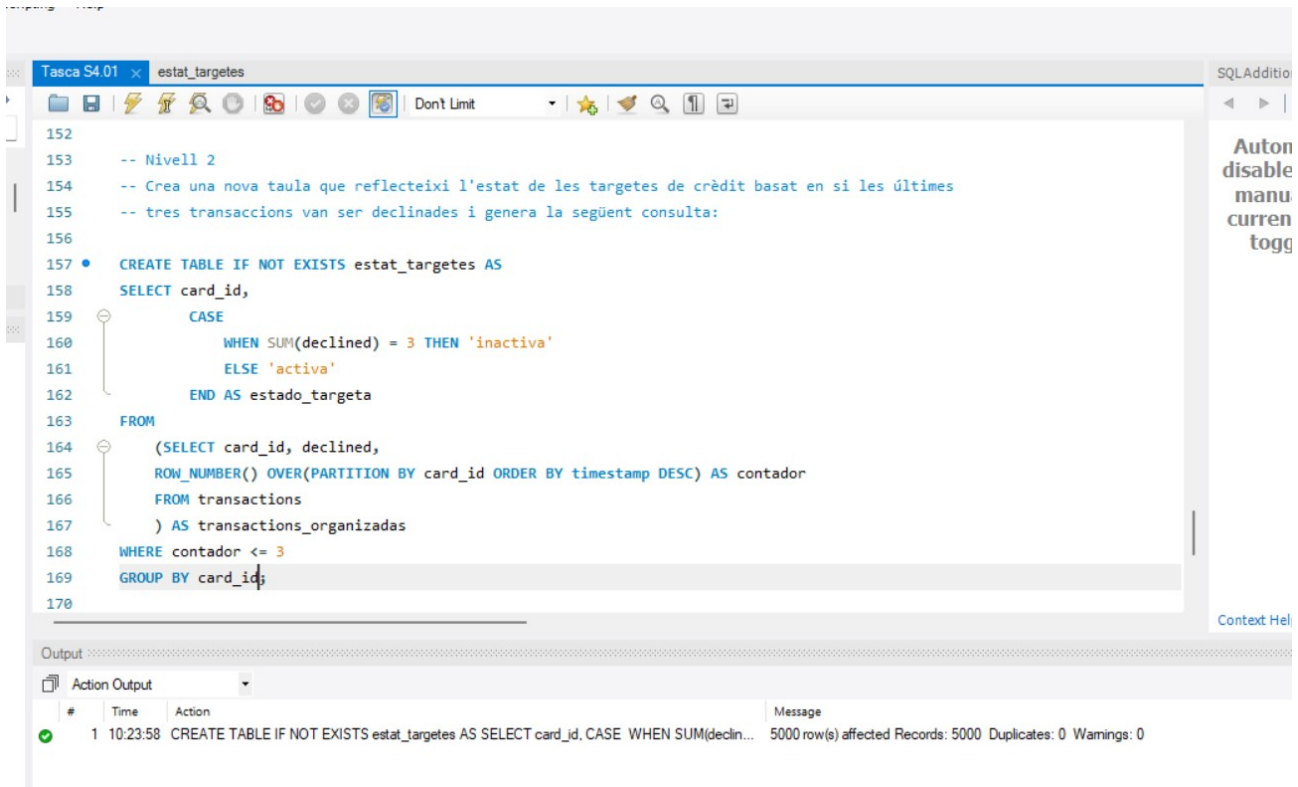
Below the query editor, the 'Result Grid' shows the results of the query. The grid has three columns: 'iban', 'company_name', and 'media_amount'. The results are as follows:

iban	company_name	media_amount
XX383017813919620199366352	Donec Ltd	680.69
XX637706357397570394973913	Donec Ltd	680.01
XX971393971465292202312259	Donec Ltd	645.46
XX171847116928892375969307	Donec Ltd	628.89
XX225424638818542406223575	Donec Ltd	608.68

At the bottom, the 'Output' pane shows the execution message: '1 11:50:42 SELECT cc.iban, c.company_name, ROUND(AVG(t.amount), 2) AS media_amount FROM transa... 370 row(s) returned'.

•Nivell 2

•Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades i genera la següent consulta:



```
152
153 -- Nivell 2
154 -- Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes
155 -- tres transaccions van ser declinades i genera la següent consulta:
156
157 • CREATE TABLE IF NOT EXISTS estat_targetes AS
158   SELECT card_id,
159          CASE
160             WHEN SUM(declined) = 3 THEN 'inactiva'
161             ELSE 'activa'
162          END AS estado_targeta
163   FROM
164      (SELECT card_id, declined,
165         ROW_NUMBER() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS contador
166       FROM transactions
167      ) AS transactions_organizadas
168   WHERE contador <= 3
169   GROUP BY card_id;
```

Output

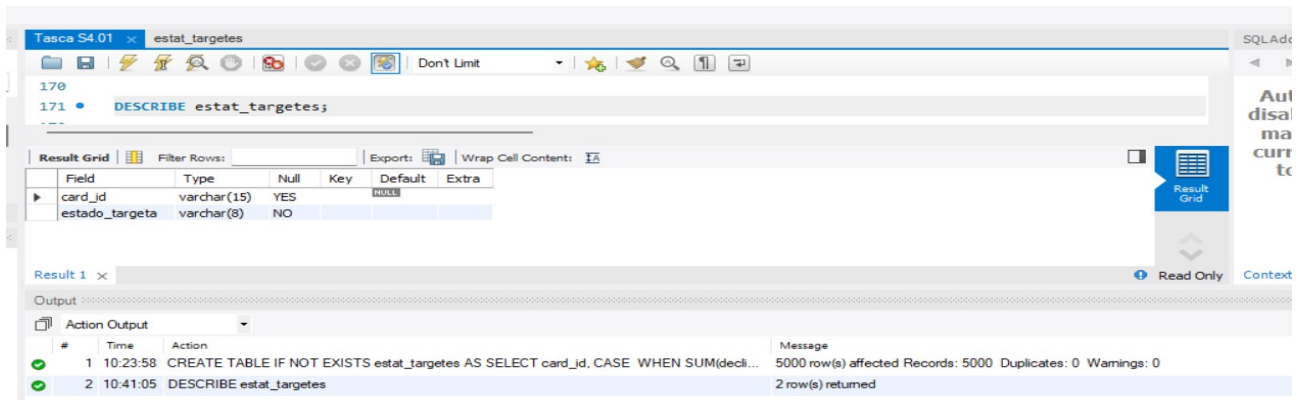
#	Time	Action	Message
1	10:23:58	CREATE TABLE IF NOT EXISTS estat_targetes AS SELECT card_id, CASE WHEN SUM(declin...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

- Se nos solicita crear una nueva tabla llamada estat_targetes para ello en el SELECT pondremos la columna card_id de la tabla transactions y mediante la expresión CASE generaremos otra columna llamada estado_targeta que tal y como nos indica si la suma de declined es igual a 3 la targeta estará inactiva y en caso contrario estará activa.

Posteriormente hacemos un FROM con una subconsulta para crear una tabla temporal con una función de ventana ROW_NUMBER() OVER(PARTITION BY credit_card ORDER BY timestamp DESC) creando una columna que le llamaremos contador, en este caso la función nos recorrera todas las transactions que se han realizado con cada credit_card y con el ORDER BY en orden descendente nos las enumerara en el contador la 1ª la mas actual y asi sucesivamente con cada credit_card realizamos el FROM transactions y renombramos la tabla temporal de la subconsulta como transactions_organizadas.

Le ponemos la clausala WHERE para indicarle al contador que solo nos coja la 1,2 y 3 las tres últimas transactions y finalmente la agruparemos por el card_id. Ejecutamos el código y obtenemos la nueva tabla estat_targetes.

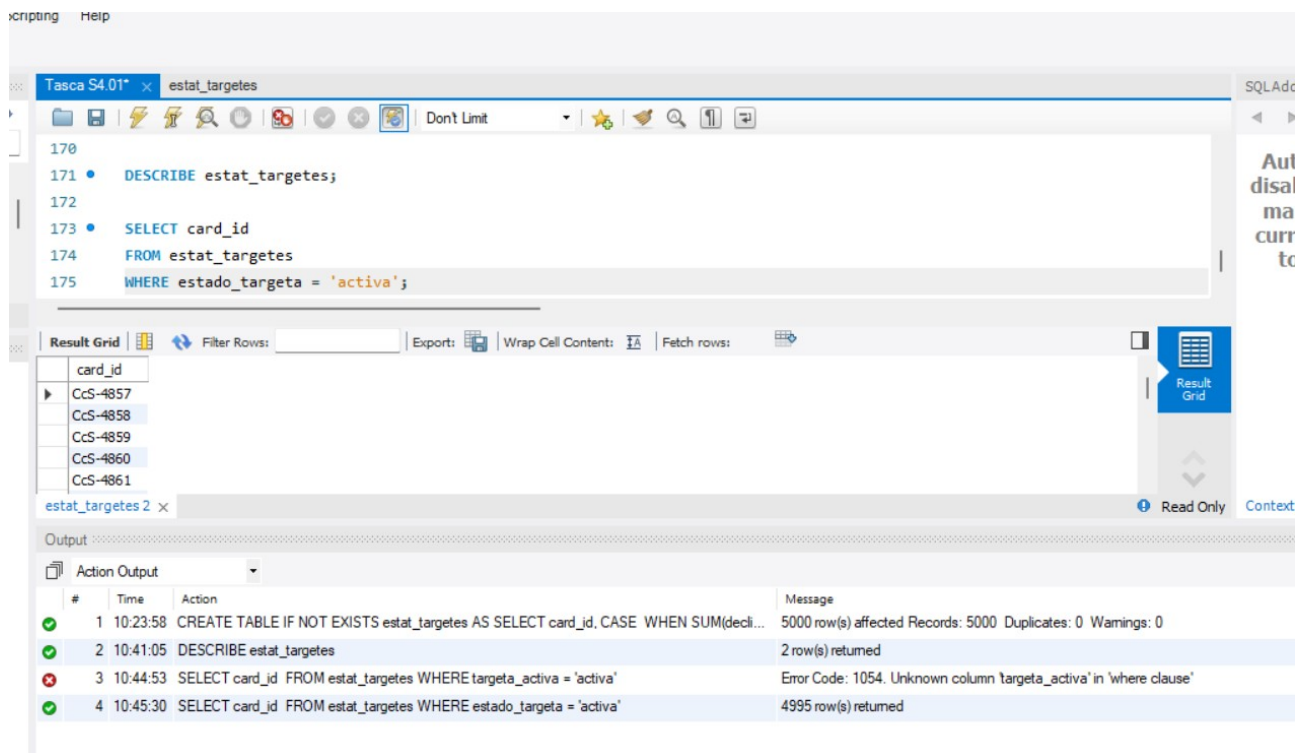
- Realizamos un DESCRIBE estat_targetes para visualizar como queda la tabla.



- Realizaremos la siguiente consulta que se nos pide en el siguiente enunciado.

·Exercici 1

·Quantes targetes estan actives?



- Finalmente obtenemos que las targetas activas son 4995 y las inactivas son 5 las cuales tienen las 3 ultimas transactions con declined = 1.

- Realizamos la conexión de la nueva tabla estat_targetes con credit_card

```

171 • DESCRIBE estat_targetes;
172
173 • SELECT card_id
174 FROM estat_targetes
175 WHERE estado_targeta = 'activa';
176
177 -- Unimos la tabla estat_targetes con la tabla credit_card
178 • ALTER TABLE estat_targetes
179 ADD CONSTRAINT fk_estat_targetes_credit_card
180 FOREIGN KEY (card_id)
181 REFERENCES credit_card(id);
182
183
184 -- Nivell 3

```

Output

#	Time	Action	Message
1	11:58:40	ALTER TABLE estat_targetes ADD CONSTRAINT fk_estat_targetes_credit_card FOREIGN KEY ...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

·Nivell 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. Genera la següent consulta:

- Primero de todo creamos la tabla products con todas sus variables y el tipo de dato que le asignamos a cada variable.

```

181 -- Creamos la tabla products
182 • CREATE TABLE IF NOT EXISTS products (
183     id INT PRIMARY KEY AUTO_INCREMENT,
184     product_name VARCHAR(100),
185     price VARCHAR(20),
186     colour VARCHAR(30),
187     weight DECIMAL(10,2),
188     warehouse_id VARCHAR(20)
189 );
190
191 • DESCRIBE products;

```

Result Grid

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
product_name	varchar(100)	YES		NULL	
price	varchar(20)	YES		NULL	
colour	varchar(30)	YES		NULL	
weight	decimal(10,2)	YES		NULL	
warehouse_id	varchar(20)	YES		NULL	

Result 1 x

Output

#	Time	Action	Message
1	11:07:15	CREATE TABLE IF NOT EXISTS products (id INT PRIMARY KEY AUTO_INCREMENT, prod...	0 row(s) affected
2	11:07:26	DESCRIBE products	6 row(s) returned

- Cargamos los datos de PRODUCTS mediante el archivo .csv que tenemos y comprobamos que se hayan cargado correctamente.

The screenshot shows the SQL Developer interface with a task named 'estad_targetes'. The SQL editor contains the following commands:

```
193 -- cargamos lo datos del csv de products
194
195 • LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv'
196 INTO TABLE products
197 FIELDS TERMINATED BY ','
198 OPTIONALLY ENCLOSED BY ''
199 IGNORE 1 ROWS;
200 -- comprobamos que los datos se hayan cargado correctamente.
201 • SELECT*
202 FROM products;
```

The 'Result Grid' displays the following data:

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	\$161.11	#7c7c7c	1.00	WH-4
2	Tarly Stark	\$9.24	#919191	2.00	WH-3
3	duel tourney Lannister	\$171.13	#d8d8d8	1.50	WH-2
4	warden south duel	\$71.89	#111111	3.00	WH-1
5	skywalker ewok	\$171.22	#bdbdbd	3.20	WH-0
6	dooku solo	\$136.60	#c4c4c4	0.80	WH--1
7	north of Casterly	\$63.33	#b7b7b7	0.60	WH--2
8	Winterfell	\$32.37	#383838	1.40	WH--3

The 'Action Output' pane shows the following results:

#	Time	Action	Message
1	11:07:15	CREATE TABLE IF NOT EXISTS products (id INT PRIMARY KEY AUTO_INCREMENT, prod...	0 row(s) affected
2	11:07:26	DESCRIBE products	6 row(s) returned
3	11:10:14	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv' I...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
4	11:11:25	SELECT* FROM products	100 row(s) returned

- Mediante el siguiente código desactivamos la protección que impide realizar actualizaciones sin el uso de la clausula where.

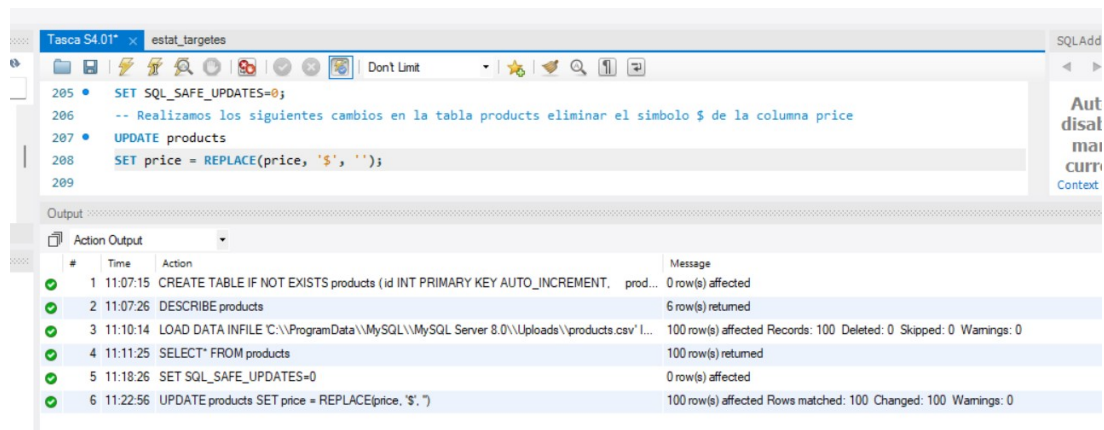
The screenshot shows the SQL Developer interface with a task named 'estad_targetes'. The SQL editor contains the following commands:

```
199 IGNORE 1 ROWS;
200 -- comprobamos que los datos se hayan cargado correctamente.
201 • SELECT*
202 FROM products;
203
204 -- Desactivamos la protección para poder realizar cambios necesarios en la tabla products
205 • SET SQL_SAFE_UPDATES=0;
206 --
```

The 'Action Output' pane shows the following results:

#	Time	Action	Message
1	11:07:15	CREATE TABLE IF NOT EXISTS products (id INT PRIMARY KEY AUTO_INCREMENT, prod...	0 row(s) affected
2	11:07:26	DESCRIBE products	6 row(s) returned
3	11:10:14	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv' I...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
4	11:11:25	SELECT* FROM products	100 row(s) returned
5	11:18:26	SET SQL_SAFE_UPDATES=0	0 row(s) affected

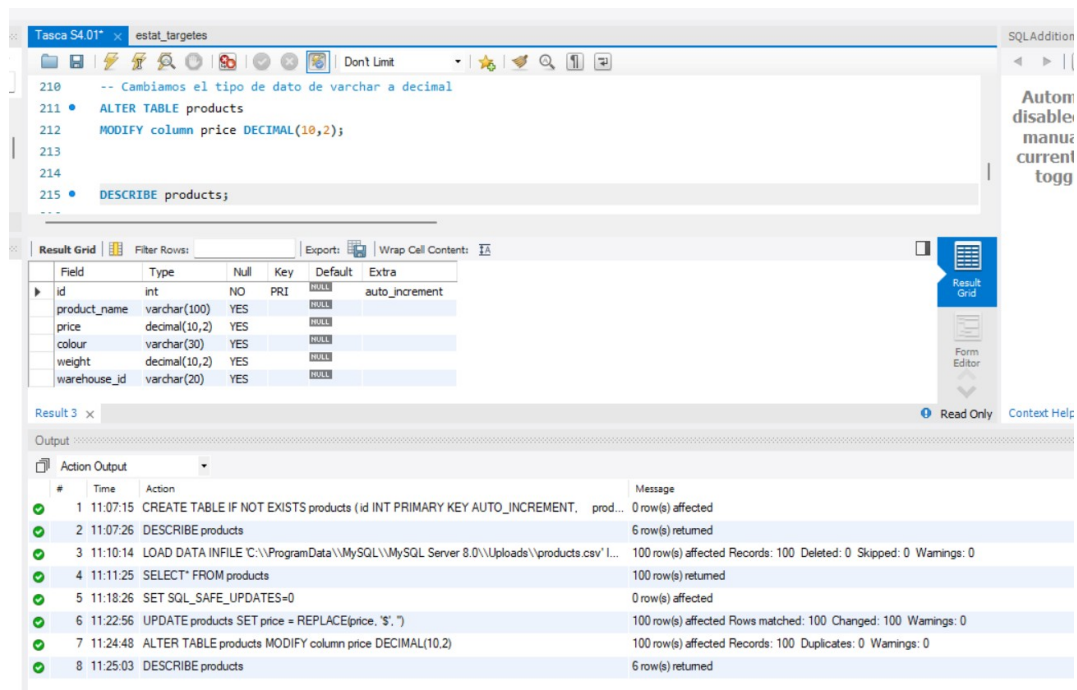
- Realizaremos los siguientes cambios , eliminaremos el simbolo del dollar y cambiaremos el tipo de dato de la columna price de VARCHAR a DECIMAL y comprobamos que los cambios estén realizados. (modificar a la hora de cargar la tabla products)



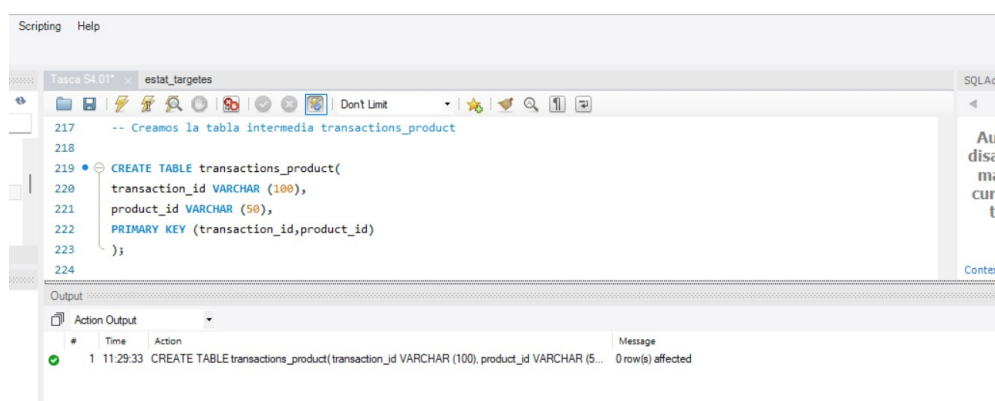
- Volvemos a habilitar la protección mediante la siguiente instrucción

SET SQL_SAFE_UPDATES = 1;

- Modificamos el tipo de dato en la columna price a decimal



- Creamos la tabla intermedia que llamaremos transactions_product.



- Cargamos los datos de la tabla transactions_product mediante el siguiente código.
- El INSERT INTO es para cargar los datos
- Con el SELECT seleccionamos pares de Id de transactions con Id de products a insertar en la nueva tabla donde cada id transactions tiene que tener un id products si coincide con product_ids de la tabla transacciones.
- Realizamos una unión entre las tablas transactions y products y en el ON realizamos una función de cadena FIND_IN_SET donde buscaremos que coincidan el id de products con el product_ids de la tabla transactions el REPLACE se utiliza en este caso para eliminar los espacios después de la coma y si coinciden se ejecuta el JOIN y se carga en la nueva tabla.

The screenshot shows a SQL IDE window with a script titled 'estat_targetes'. The script contains the following SQL statements:

```

225 -- Insertamos los datos en la tabla transactions_product
226
227 • INSERT INTO transactions_product (transaction_id, product_id)
228   SELECT t.id AS transaction_id, p.id AS product_id
229     FROM transactions t
230     JOIN products p
231     ON FIND_IN_SET( p.id , REPLACE(t.product_ids, ' ', '')) > 0;
232

```

The Output pane shows the execution results:

#	Time	Action	Message
1	11:29:33	CREATE TABLE transactions_product(transaction_id VARCHAR(100), product_id VARCHAR(5...)	0 row(s) affected
2	11:43:38	INSERT INTO transactions_product (transaction_id, product_id) SELECT t.id AS transaction_id, p...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

- Realizamos la unión entre credit_card y estat_targetes

The screenshot shows a SQL IDE window with a script titled 'estat_targetes'. The script contains the following SQL statements:

```

171 • DESCRIBE estat_targetes;
172
173 • SELECT card_id
174   FROM estat_targetes
175  WHERE estado_targeta = 'activa';
176
177 -- Unimos la tabla estat_targetes con la tabla credit_card
178 • ALTER TABLE estat_targetes
179   ADD CONSTRAINT fk_estat_targetes_credit_card
180   FOREIGN KEY (card_id)
181   REFERENCES credit_card(id);
182
183
184 -- Nivell 3

```

The Output pane shows the execution results:

#	Time	Action	Message
1	11:58:40	ALTER TABLE estat_targetes ADD CONSTRAINT fk_estat_targetes_credit_card FOREIGN KEY ...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

- Generamos la relación de la tabla transactions_product con transactions

```
232
233 -- Uniremos las tablas transactions con transactions_product y transactions_product con products.
234
235 • ALTER TABLE transactions_product
236   ADD CONSTRAINT fk_transactions_product_transactions
237   FOREIGN KEY (transaction_id)
238   REFERENCES transactions(id);
239
```

Output

#	Time	Action	Message
1	12:34:50	ALTER TABLE transactions_product ADD CONSTRAINT fk_transactions_product_transactions F...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

- Generamos la relación de la tabla transactions_product con products

- Para ello tenemos que modificar el tipo de dato de la columna product_id de la tabla transactions_product de VARCHAR a INT

```
239
240 -- para poder unir la tabla transactions_product con products necesito cambiar el tipo de dato de la columna product_id
241
242 • ALTER TABLE transactions_product
243   MODIFY COLUMN product_id INT;
244
```

Output

#	Time	Action	Message
1	12:34:50	ALTER TABLE transactions_product ADD CONSTRAINT fk_transactions_product_transactions F...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	12:39:40	ALTER TABLE transactions_product MODIFY COLUMN product_id INT	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0

- Generamos la relación

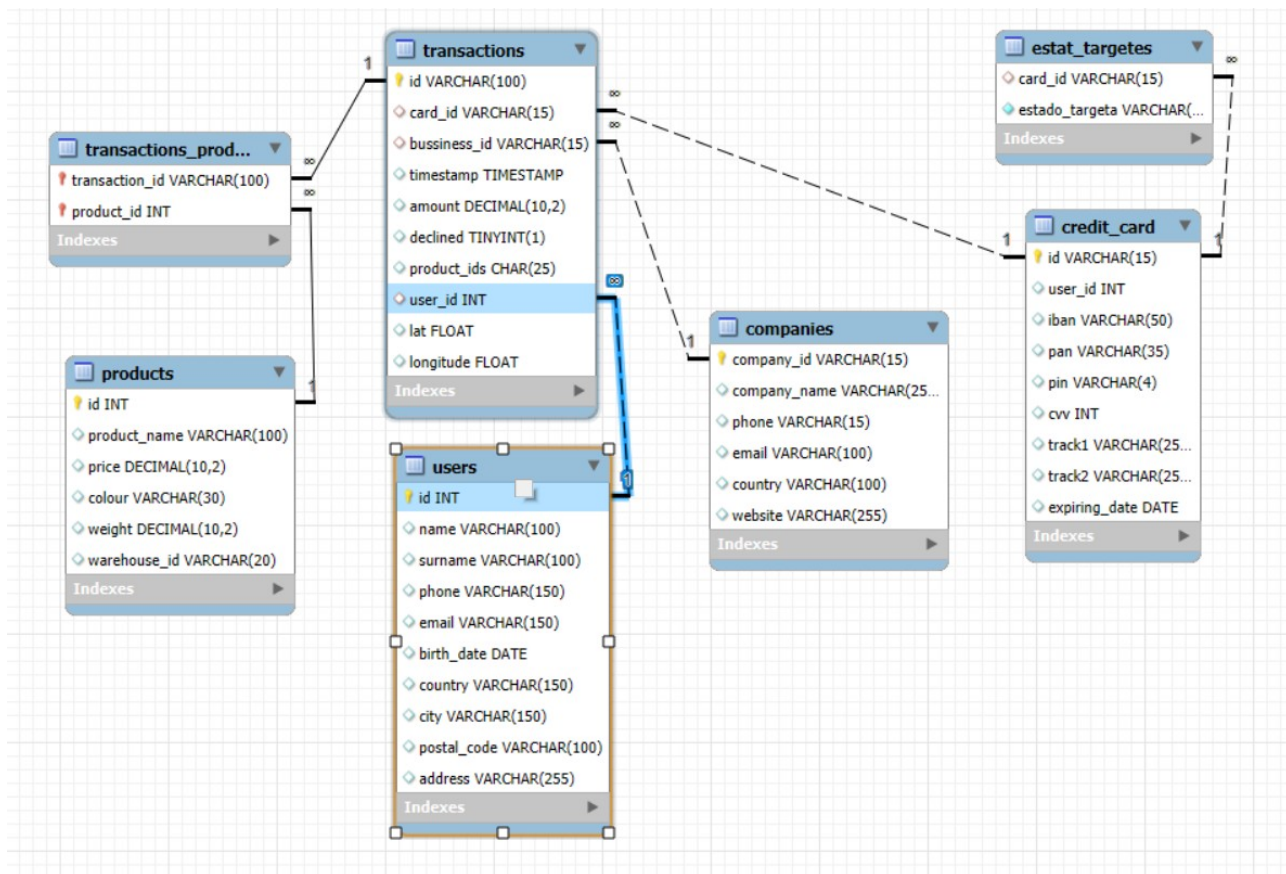
entre las tablas

```
245 -- Unimos la tabla transactions_product con la tabla products
246 • ALTER TABLE transactions_product
247   ADD CONSTRAINT fk_transactions_product_products
248   FOREIGN KEY (product_id)
249   REFERENCES products(id);
250
```

Output

#	Time	Action	Message
1	12:34:50	ALTER TABLE transactions_product ADD CONSTRAINT fk_transactions_product_transactions F...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
2	12:39:40	ALTER TABLE transactions_product MODIFY COLUMN product_id INT	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0
3	12:41:37	ALTER TABLE transactions_product ADD CONSTRAINT fk_transactions_product_products FOR...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

- Visualización del EER del diagrama de la base de datos



- Este diagrama representaría un modelado en copo de nieve ya que no todas las tablas se relacionan directamente con la tabla de hechos transactions, tenemos una tabla intermedia transactions_product entre products y transactions con una relación fuerte entre ambas tablas. Las tablas company, users y credit_cards tienen una relación débil con la tabla de hechos transactions ya que su relación es no identificadora, eso quiere decir que transactions tiene su propia clave primaria independiente de sus claves foraneas bussiness_id, card_id y user_id por eso se representa la relación con una línea discontinua, eso quiere decir que la existencia de una transaction no se identifica exclusivamente ni con la empresa, ni con la tarjeta y ni con el usuario a la que pertenece. Y la tabla estat_targetes se realaciona con credit_card mediante la primary key id de credit_card con la fk card_id

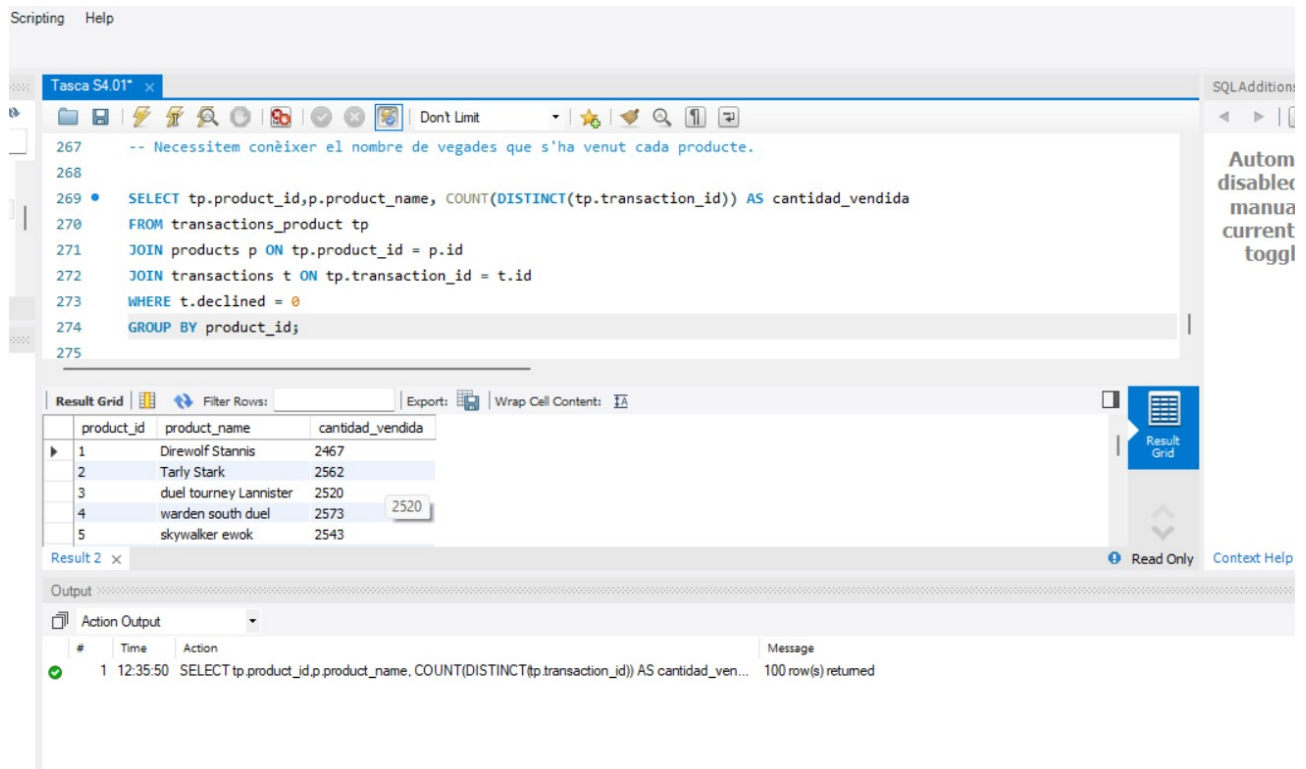
- La cardinalidad del diagrama sería la siguiente.

- Companies(dimensión) a Transactions(hechos) de 1:N
- Users(dimensión) a Transactions(hechos) de 1:N
- Credit_card(dimensión) a Transactions(hechos) de 1:N
- Credit_card(dimensión) a estat_targetes(subdimensión) de 1:N
- Products(dimensión) a Transactions(hechos) de N:M por eso creamos una tabla intermedia con 2 llaves foraneas que a su vez son primary key compuestas que son las claves primarias de Products y Transactions y obtenemos la siguiente relación.
- Products(dimensión) a Transactions_product(dimensión) de 1:N
- transactions(hechos) a Transactions_product(dimensión) de 1:N

•Exercici 1

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

- Para realizar este ejercicio realizamos el siguiente código.



The screenshot shows a SQL IDE interface with a script editor and a results grid. The script editor contains the following SQL query:

```
267 -- Necessitem conèixer el nombre de vegades que s'ha venut cada producte.
268
269 • SELECT tp.product_id,p.product_name, COUNT(DISTINCT(tp.transaction_id)) AS cantidad_vendida
270 FROM transactions_product tp
271 JOIN products p ON tp.product_id = p.id
272 JOIN transactions t ON tp.transaction_id = t.id
273 WHERE t.declined = 0
274 GROUP BY product_id;
275
```

The results grid displays the following data:

	product_id	product_name	cantidad_vendida
▶	1	Direwolf Stannis	2467
	2	Tarly Stark	2562
	3	duel tourney Lannister	2520
	4	warden south duel	2573
	5	skywalker ewok	2543

The output pane shows the following message:

```
1 12:35:50 SELECT tp.product_id,p.product_name, COUNT(DISTINCT(tp.transaction_id)) AS cantidad_ven... 100 row(s) returned
```

- En esta consulta en el SELECT le solicitamos el product_id de la tabla transactions_product el product_name de la tabla products y un COUNT(DISTINCT) de transaction_id de la tabla transactions_product que le llamaremos cantidad_vendida realizamos 2 joins entre las tablas transactions_product y products y transactions_product con transactions le poenemos la condición de declined de la tabla transactions sea igual a 0 y los agrupamos con el GROUP BY por el product_id de la tabla transactions_product, y finalmente obtenemos el resultado que es cuantaas veces se ha vendido cada producto.