

Tasca S4.01. Creació de bases de dades

Nivell 1

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

Creación de la base de datos

- Creamos una base de datos llamada "operativa".
- La seleccionamos para trabajar con ella.

```
CREATE DATABASE operativa;  
  
-- Seleccionamos la base de datos  
USE operativa;
```

```

1  -- Creamos la base de datos
2  • CREATE DATABASE operativa;
3
4  • USE operativa;
5
6

```

Output

#	Time	Action	Message
1	10:08:49	CREATE DATABASE operativa	1 row(s) affected
2	10:08:56	USE operativa	0 row(s) affected

Creación de las tablas

- Creamos la tabla "companies"
 - Identificamos el campo "company_id" como Primary Key o PK
 - Marcamos el campo "email" como UNIQUE al tratarse de un campo único que no puede tener duplicados, ya que cada dirección de correo debe ser única. Además, al indexarse automáticamente también optimizamos el rendimiento de la base de datos al realizar búsquedas a través de estos campos.
 - El campo "phone" también podría ser un campo UNIQUE, pero al haber priorizado el campo anterior no es recomendable tener varios INDEX en la misma tabla al poder verse afectada la optimización de la base de datos.

```

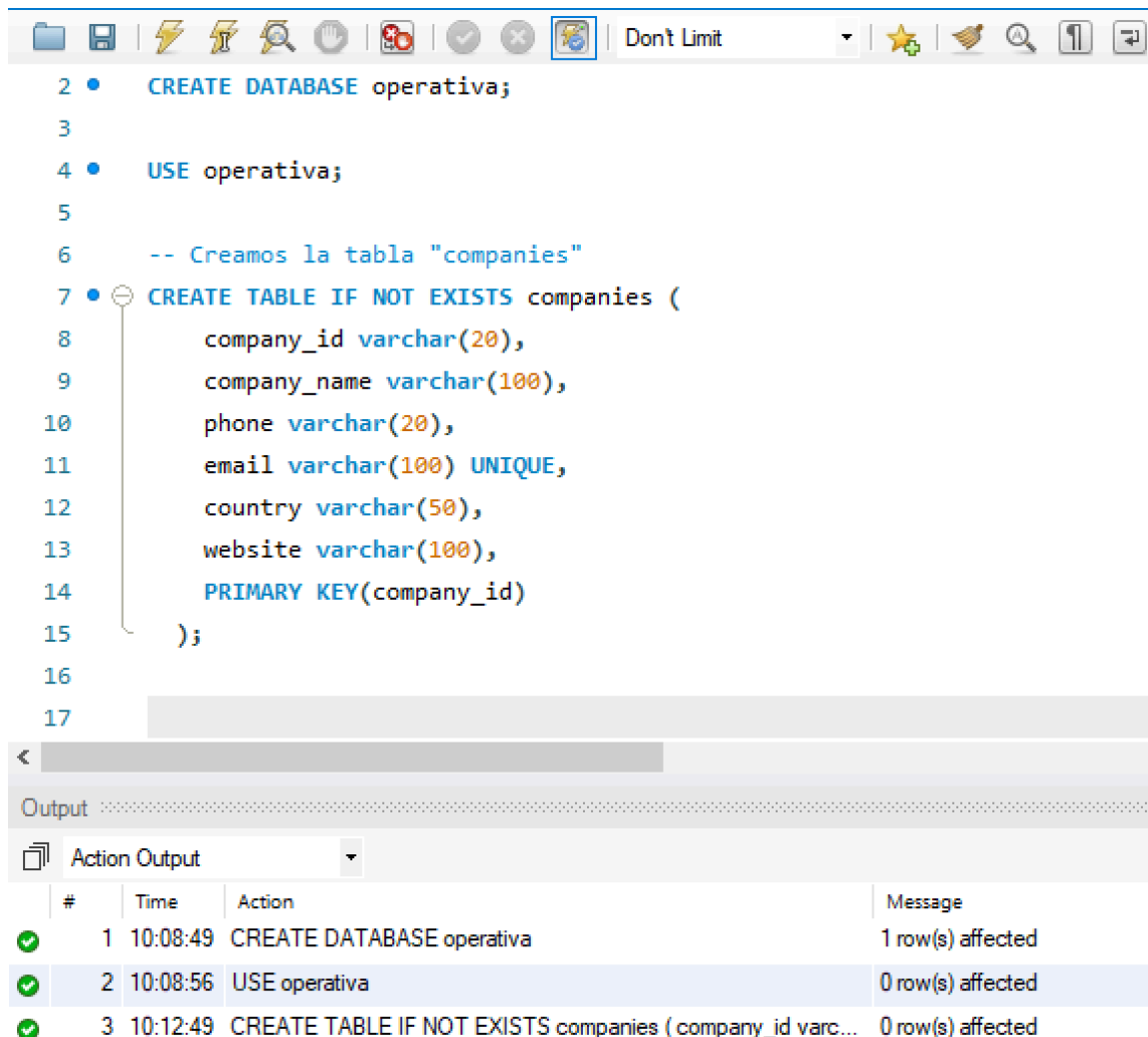
CREATE TABLE IF NOT EXISTS companies (
  company_id varchar(20),
  company_name varchar(100),
  phone varchar(20),
  email varchar(100) UNIQUE,

```

```

country varchar(50),
website varchar(100),
PRIMARY KEY(company_id)
);

```



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

```

2 • CREATE DATABASE operativa;
3
4 • USE operativa;
5
6 -- Creamos la tabla "companies"
7 • CREATE TABLE IF NOT EXISTS companies (
8     company_id varchar(20),
9     company_name varchar(100),
10    phone varchar(20),
11    email varchar(100) UNIQUE,
12    country varchar(50),
13    website varchar(100),
14    PRIMARY KEY(company_id)
15 );
16
17

```

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

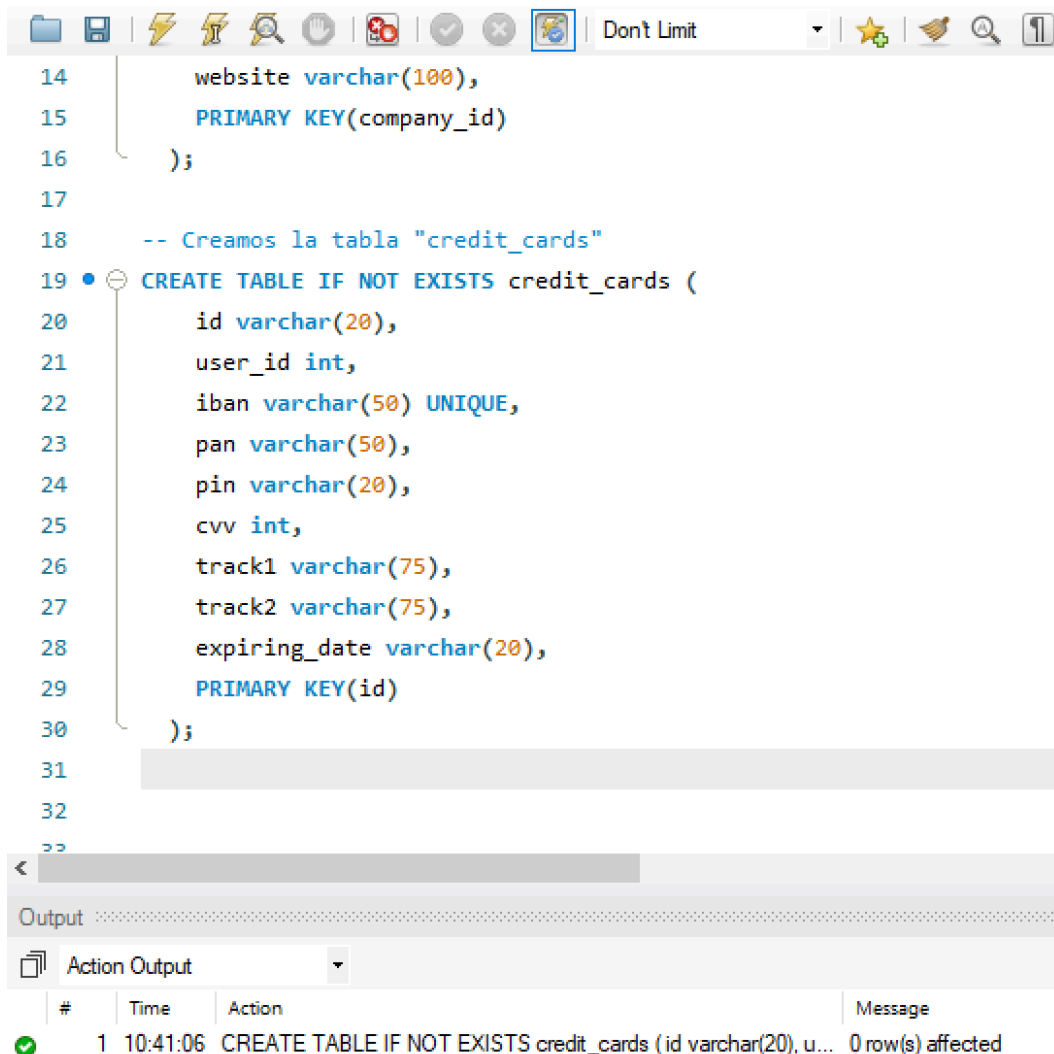
#	Time	Action	Message
✓ 1	10:08:49	CREATE DATABASE operativa	1 row(s) affected
✓ 2	10:08:56	USE operativa	0 row(s) affected
✓ 3	10:12:49	CREATE TABLE IF NOT EXISTS companies (company_id varc...	0 row(s) affected

- Creamos la tabla "credit_cards"
 - Identificamos "id" como Primary Key de la tabla
 - Marcamos el campo iban como UNIQUE

```

CREATE TABLE IF NOT EXISTS credit_cards (
    id varchar(20),
    user_id int,
    iban varchar(50) UNIQUE,
    pan varchar(50),
    pin varchar(20),
    cvv int,
    track1 varchar(75),
    track2 varchar(75),
    expiring_date varchar(20),
    PRIMARY KEY(id)
);

```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and search. The main editor displays SQL code with line numbers 14 through 33. The code defines a table 'credit_cards' with various fields and constraints. Below the editor, the 'Output' pane is visible, showing the 'Action Output' for the executed statement. The output table has columns for '#', 'Time', 'Action', and 'Message'. A single row indicates the successful execution of the CREATE TABLE statement at 10:41:06, with 0 rows affected.

```

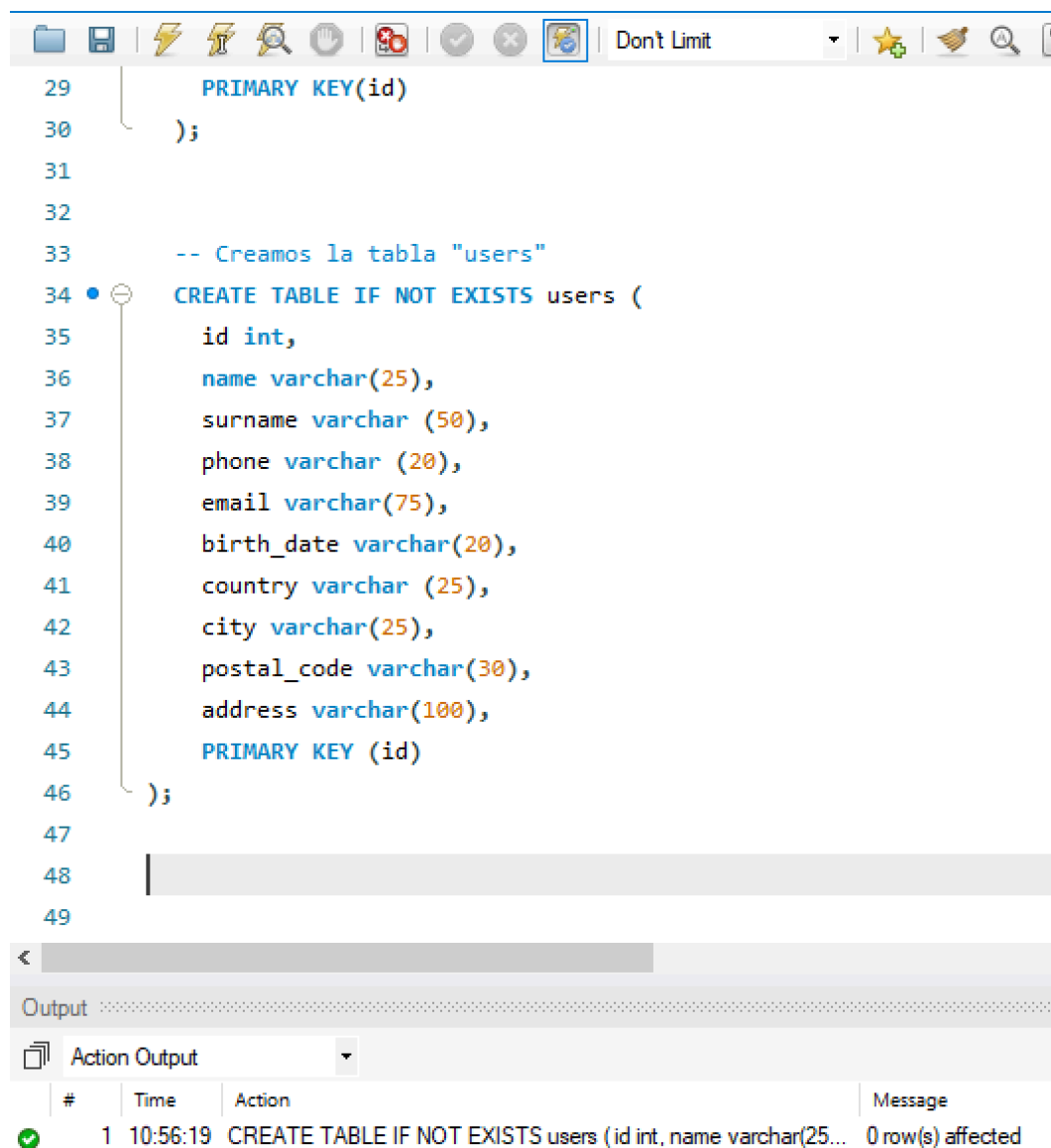
14     website varchar(100),
15     PRIMARY KEY(company_id)
16 );
17
18 -- Creamos la tabla "credit_cards"
19 • CREATE TABLE IF NOT EXISTS credit_cards (
20     id varchar(20),
21     user_id int,
22     iban varchar(50) UNIQUE,
23     pan varchar(50),
24     pin varchar(20),
25     cvv int,
26     track1 varchar(75),
27     track2 varchar(75),
28     expiring_date varchar(20),
29     PRIMARY KEY(id)
30 );
31
32
33

```

#	Time	Action	Message
✓ 1	10:41:06	CREATE TABLE IF NOT EXISTS credit_cards (id varchar(20), u...	0 row(s) affected

- Creamos la tabla "users"
 - Tenemos 3 archivos .csv de usuarios diferentes y cada uno corresponde a un país diferente (Canada, United Kingdom y United States). Al conservar la misma estructura de columnas, decidimos crear una única tabla en la cual insertaremos los datos de los tres archivos en lugar de crear tres tablas diferentes.
 - Asignamos el campo "id" como Primary Key.

```
CREATE TABLE IF NOT EXISTS users (  
  id int,  
  name varchar(25),  
  surname varchar (50),  
  phone varchar (20),  
  email varchar(75),  
  birth_date varchar(20),  
  country varchar (25),  
  city varchar(25),  
  postal_code varchar(30),  
  address varchar(100),  
  PRIMARY KEY (id)  
);
```



```
29     PRIMARY KEY(id)
30 );
31
32
33 -- Creamos la tabla "users"
34 • CREATE TABLE IF NOT EXISTS users (
35     id int,
36     name varchar(25),
37     surname varchar (50),
38     phone varchar (20),
39     email varchar(75),
40     birth_date varchar(20),
41     country varchar (25),
42     city varchar(25),
43     postal_code varchar(30),
44     address varchar(100),
45     PRIMARY KEY (id)
46 );
47
48
49
```

Output

Action Output

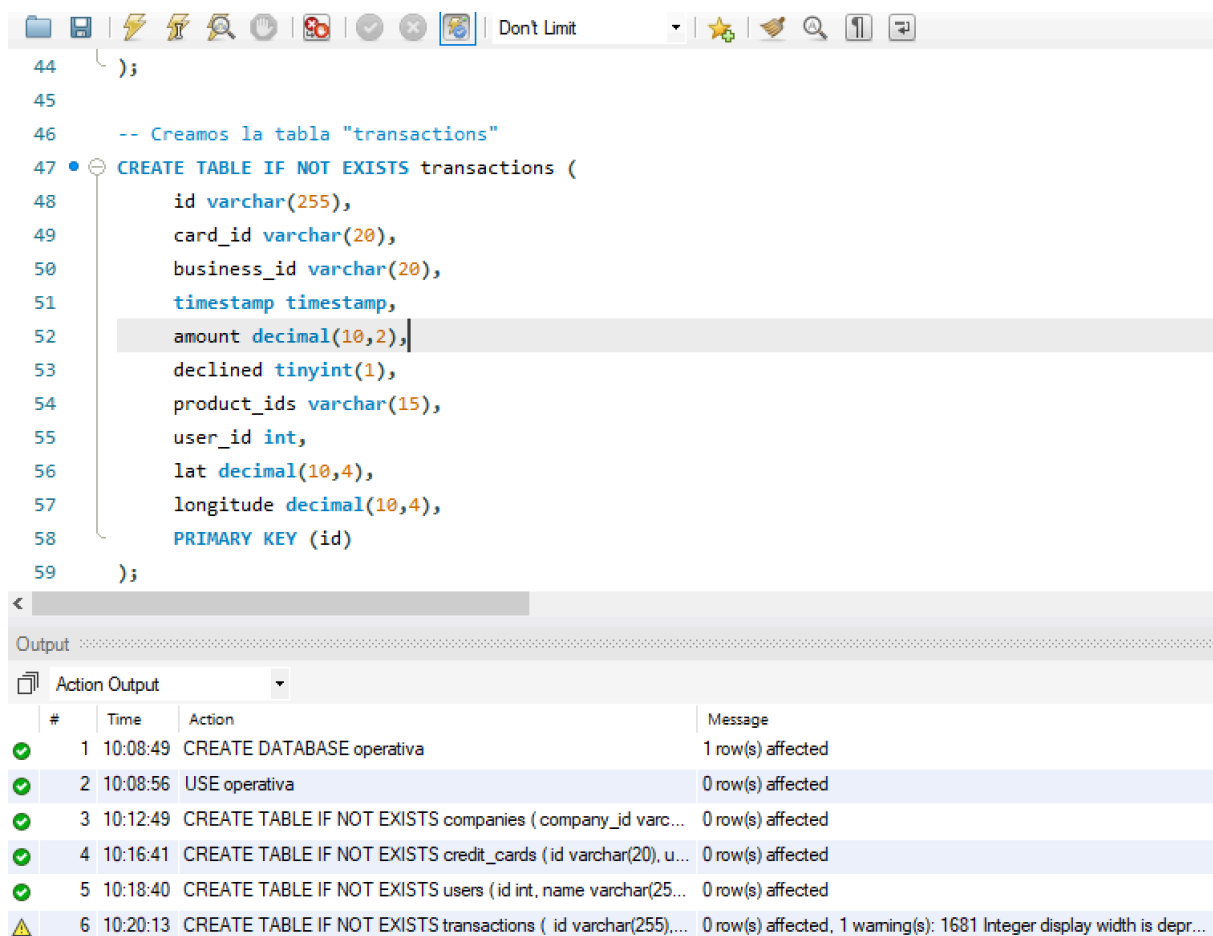
#	Time	Action	Message
✓ 1	10:56:19	CREATE TABLE IF NOT EXISTS users (id int, name varchar(25...	0 row(s) affected

- Creamos la tabla "transactions"
 - Asignamos el campo "id" como Primary Key
 - Al tratarse de la tabla de hechos de la base de datos, creamos relaciones con las otras tablas mediante diversas Foreign Key, una por cada tabla de dimensiones, que las vincula con su Primary Key en cada una de esas tablas.

```

CREATE TABLE IF NOT EXISTS transactions (
    id varchar(255),
    card_id varchar(20),
    business_id varchar(20),
    timestamp timestamp,
    amount decimal(10,2),
    declined tinyint(1),
    product_ids varchar(15),
    user_id int,
    lat decimal(10,4),
    longitude decimal(10,4),
    PRIMARY KEY (id)
);

```



The screenshot shows a database IDE with a toolbar at the top. The SQL editor contains the following code:

```

44 );
45
46 -- Creamos la tabla "transactions"
47 CREATE TABLE IF NOT EXISTS transactions (
48     id varchar(255),
49     card_id varchar(20),
50     business_id varchar(20),
51     timestamp timestamp,
52     amount decimal(10,2),
53     declined tinyint(1),
54     product_ids varchar(15),
55     user_id int,
56     lat decimal(10,4),
57     longitude decimal(10,4),
58     PRIMARY KEY (id)
59 );

```

Below the editor is an "Output" window showing the "Action Output". It contains a table with the following data:

#	Time	Action	Message
✓ 1	10:08:49	CREATE DATABASE operativa	1 row(s) affected
✓ 2	10:08:56	USE operativa	0 row(s) affected
✓ 3	10:12:49	CREATE TABLE IF NOT EXISTS companies (company_id varc...	0 row(s) affected
✓ 4	10:16:41	CREATE TABLE IF NOT EXISTS credit_cards (id varchar(20), u...	0 row(s) affected
✓ 5	10:18:40	CREATE TABLE IF NOT EXISTS users (id int, name varchar(25...	0 row(s) affected
⚠ 6	10:20:13	CREATE TABLE IF NOT EXISTS transactions (id varchar(255),...	0 row(s) affected, 1 warning(s): 1681 Integer display width is depr...

Activación de la carga local

Para poder cargar un archivo .csv en local, hemos de verificar si la opción de carga local está activada. Al realizar la comprobación, aparece como OFF, por lo que procederemos a su activación

```
SHOW VARIABLES LIKE "local_infile"; -- Comprobamos si la carga local está activa  
  
SET GLOBAL local_infile=1; -- Solo en caso que estuviera en OFF
```

	Variable_name	Value
▶	local_infile	ON

Result 2 x				
Output				
Action Output				
#	Time	Action	Message	
✓ 1	20:35:38	SHOW VARIABLES LIKE "local_infile"	1 row(s) returned	

Ahora tenemos la carga local activada a nivel general, pero también hemos de activarla en la propia base de datos y en MySQL Server.

Activación en la base de datos

1. En el menú superior de Workbench, vamos a "DATABASE" y después a "MANAGE CONNECTIONS"
2. Seleccionamos "Local Instance MySQL 80"
3. En la pestaña "advanced", añadimos la siguiente línea:
OPT_LOCAL_INFILE=1

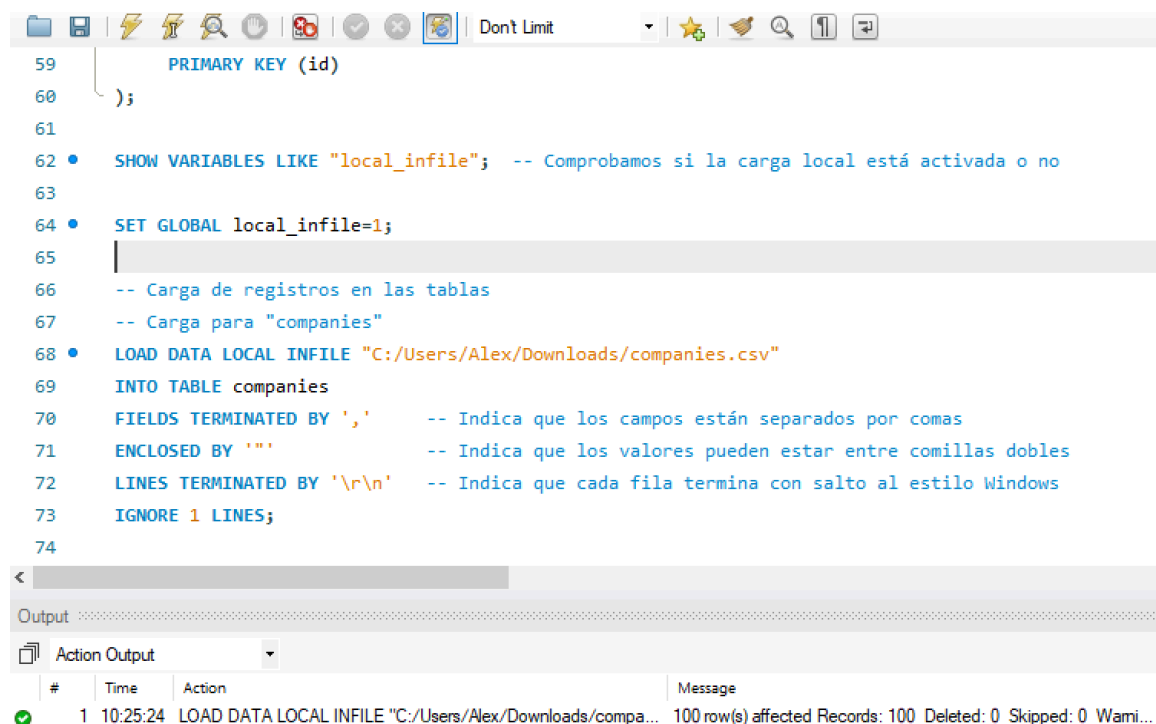
Ya podemos proceder a insertar los registros para cada una de nuestras tablas.

Carga de registros en las tablas

Carga de registros para la tabla "companies"

- Ignoramos la primera fila del .csv al tratarse de los encabezados de las columnas.

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/companies.csv"
INTO TABLE companies
FIELDS TERMINATED BY ',' -- Indica que los campos están separados por coma
ENCLOSED BY '"' -- Indica que los valores pueden estar entre comillas
LINES TERMINATED BY '\r\n' -- Indica que cada fila termina con salto al estilo Windows
IGNORE 1 LINES;
```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The main editor displays a SQL script with line numbers 59 to 74. The script defines a primary key for 'id' and then executes a 'LOAD DATA LOCAL INFILE' command to load data from 'C:/Users/Alex/Downloads/companies.csv' into the 'companies' table. The script includes comments explaining the field and line terminators and the 'IGNORE 1 LINES' clause. Below the editor, the 'Output' pane shows the 'Action Output' for the executed command, indicating that 100 rows were affected.

```
59      PRIMARY KEY (id)
60    );
61
62 • SHOW VARIABLES LIKE "local_infile"; -- Comprobamos si la carga local está activada o no
63
64 • SET GLOBAL local_infile=1;
65
66 -- Carga de registros en las tablas
67 -- Carga para "companies"
68 • LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/companies.csv"
69 INTO TABLE companies
70 FIELDS TERMINATED BY ',' -- Indica que los campos están separados por comas
71 ENCLOSED BY '"' -- Indica que los valores pueden estar entre comillas dobles
72 LINES TERMINATED BY '\r\n' -- Indica que cada fila termina con salto al estilo Windows
73 IGNORE 1 LINES;
74
```

Output

#	Time	Action	Message
✓ 1	10:25:24	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/compa...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Wami...

- Comprobamos que los datos se hayan cargado correctamente

71 ENCLOSED BY '''' -- Indica que los valores pueden estar entre comillas dobles
 72 LINES TERMINATED BY '\r\n' -- Indica que cada fila termina con salto al estilo Windows
 73 IGNORE 1 LINES;
 74
 75 • SELECT *
 76 FROM companies;
 77

Result Grid

	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/s
	b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/
	b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/s
	b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@	Germany	https://cnn.com/user/1
	b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum	New Zealand	https://netflix.com/sett
	b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.couk	Norway	https://nytimes.com/us
	b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cnn.com/one
	b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://netflix.com/sub
	b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@icloud.net	United States	https://ebay.com/sub
	b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.massa.integer@aol.net	Belgium	https://pinterest.com/s
	b-2262	Gravida Sagittis LLP	03 81 28 33 97	turpis.vitae@google.ca	Sweden	https://naver.com/site

companies 2 x

Output

Action Output

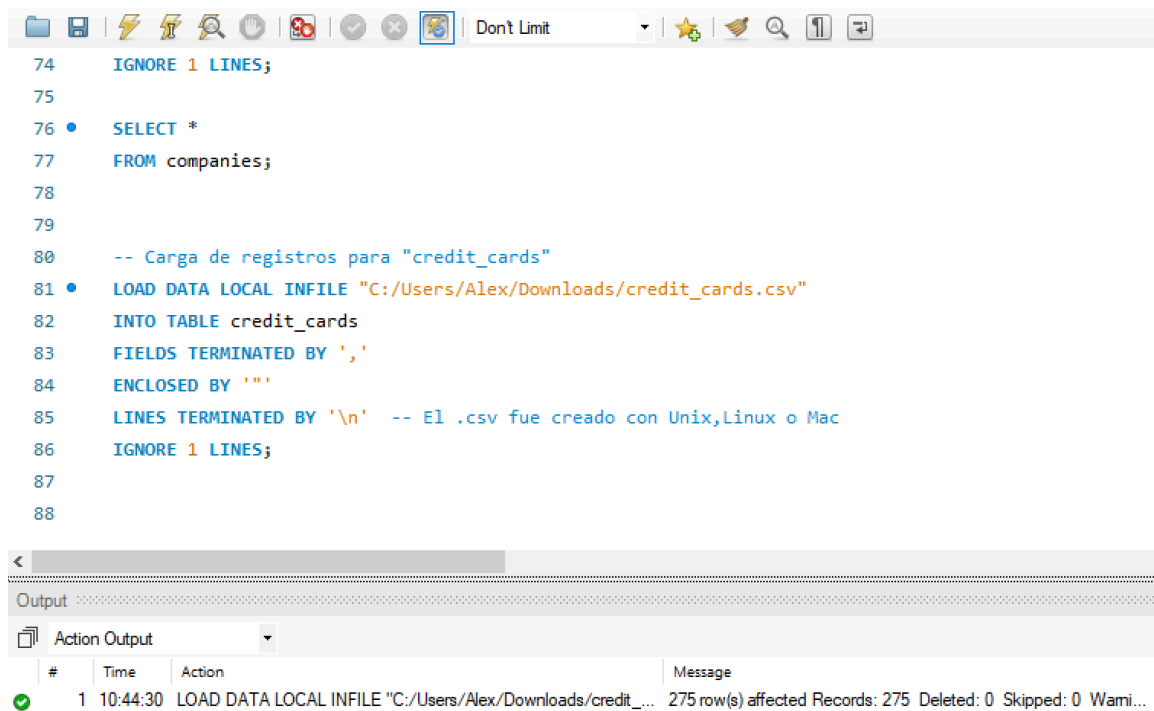
#	Time	Action	Message
✓ 1	10:25:24	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/comp...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 War...
✓ 2	10:29:05	SELECT * FROM companies	100 row(s) returned

Carga de registros para la tabla "credit_cards"

- Ignoramos la primera fila del .csv al tratarse de los encabezados de las columnas.
- Al inspeccionar el .csv con Notepad++ hemos visto que fue creado con Unix. Eso repercute en el salto de línea, por lo que en este caso sería "\n"

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/credit_cards.csv"
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY ''
```

LINES TERMINATED BY '\n' -- El .csv fue creado con Unix,Linux o Mac
IGNORE 1 LINES;



The screenshot shows a SQL IDE window with a script editor and an output pane. The script editor contains the following SQL code:

```
74 IGNORE 1 LINES;  
75  
76 • SELECT *  
77 FROM companies;  
78  
79  
80 -- Carga de registros para "credit_cards"  
81 • LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/credit_cards.csv"  
82 INTO TABLE credit_cards  
83 FIELDS TERMINATED BY ','  
84 ENCLOSED BY '"'  
85 LINES TERMINATED BY '\n' -- El .csv fue creado con Unix,Linux o Mac  
86 IGNORE 1 LINES;  
87  
88
```

The output pane shows the results of the execution:

#	Time	Action	Message
✓ 1	10:44:30	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/credit_...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Wami...

- Comprobamos que los datos se hayan cargado correctamente

83 **FIELDS TERMINATED BY** ','

84 **ENCLOSED BY** '''

85 **LINES TERMINATED BY** '\n' -- El .csv fue creado con Unix,Linux o Mac

86 **IGNORE 1 LINES;**

87

88 **SELECT ***

89 **FROM credit_cards;**

90

id	user_id	iban	pan	pin	cvv	track1
CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%88383712448554646^WovsxejDpwiev^
CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%B4621311609958661^UftuyfsSeimxn^0i
CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%B2183285104307501^CddyttcUxwfdq^
CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%B7281111956795320^XocddijBckecd^09
CcU-2966	271	BG72LKTQ15627628377363	448566 886747 7265	4900	130	%B4728932322756223^JhlgvsuFbmwgj^7
CcU-2973	270	PT87806228135092429456346	544 58654 54343 384	8760	887	%B4761405253275637^HjnnipoBlejrl^710
CcU-2980	269	DE39241881883086277136	402400 7145845969	5075	596	%B7320483593870549^OokzqxrHpsed^
CcU-2987	268	GE89681434837748781813	3763 747687 76666	2298	797	%B4750646345146674^PjmlryfGwwtrf^83
CcU-2994	267	BH62714428368066765294	344283273252593	7545	595	%B1583759784015674^GmqoyhtUtoqrn^
CcU-3001	266	CY49087426654774581266832110	511722 924833 2244	9562	867	%B6227288756728648^AwxilfcFmgvdy^2

credit_cards 9 x

Output

Action Output

#	Time	Action	Message
1	10:44:30	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/credit...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 War...
2	10:47:14	SELECT * FROM credit_cards	275 row(s) returned

Carga de registros para la tabla "users"

- Carga para "users_usa.csv"

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_usa.csv"
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '''
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```

- Carga para "users_ca.csv"

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_ca.csv"  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\r\n'  
IGNORE 1 LINES;
```

- Carga para "users_uk.csv"

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_uk.csv"  
INTO TABLE users  
FIELDS TERMINATED BY ','  
ENCLOSED BY '"'  
LINES TERMINATED BY '\r\n'  
IGNORE 1 LINES;
```

```
94 -- Carga de registros para "users"
95 • LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_usa.csv"
96 INTO TABLE users
97 FIELDS TERMINATED BY ','
98 ENCLOSED BY '"'
99 LINES TERMINATED BY '\r\n'
100 IGNORE 1 LINES;
101
102 • LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_ca.csv"
103 INTO TABLE users
104 FIELDS TERMINATED BY ','
105 ENCLOSED BY '"'
106 LINES TERMINATED BY '\r\n'
107 IGNORE 1 LINES;
108
109 • LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_uk.csv"
110 INTO TABLE users
111 FIELDS TERMINATED BY ','
112 ENCLOSED BY '"'
113 LINES TERMINATED BY '\r\n'
114 IGNORE 1 LINES;
```

Output

Action Output

#	Time	Action	Message
✓ 1	10:59:25	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Wami...
✓ 2	10:59:28	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warning...
✓ 3	10:59:30	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/users_...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warning...

- Comprobamos que los datos se han cargado correctamente

114 IGNORE 1 LINES;
 115
 116 • SELECT *
 117 FROM users;
 118
 119

	id	name	surname	phone	email	birth_date	country	city	postal_code
1	Zeus	Gamble		1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	
3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbus	56518	
4	Howard	Stafford	1-411-740-3269	ornare.egestas@cloud.edu	Feb 18, 1989	United States	Kailua	77417	
5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Sandy	31564	
6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	Nashville	96838	
7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Hillsboro	29874	
8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	Jackson	61750	
9	Mannix	Mcclain	(590) 883-2184	aliquam.nisl@outlook.com	Jan 24, 1987	United States	Richmond	35987	
10	Robert	McCarthy	(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	Eugene	85526	
11	Joan	Baird	(981) 429-8106	et@outlook.net	Feb 25, 1990	United States	Lincoln	35211	
12	Benedict	Wheeler	1-515-824-2855	tincidunt.donec.vitae@hotmail.co.uk	Aug 6, 1999	United States	Lewiston	92393	
13	Allegra	Stanton	1-927-753-6488	proin.eget@protonmail.ca	May 19, 1990	United States	Kearney	14947	

users 11 x Apply

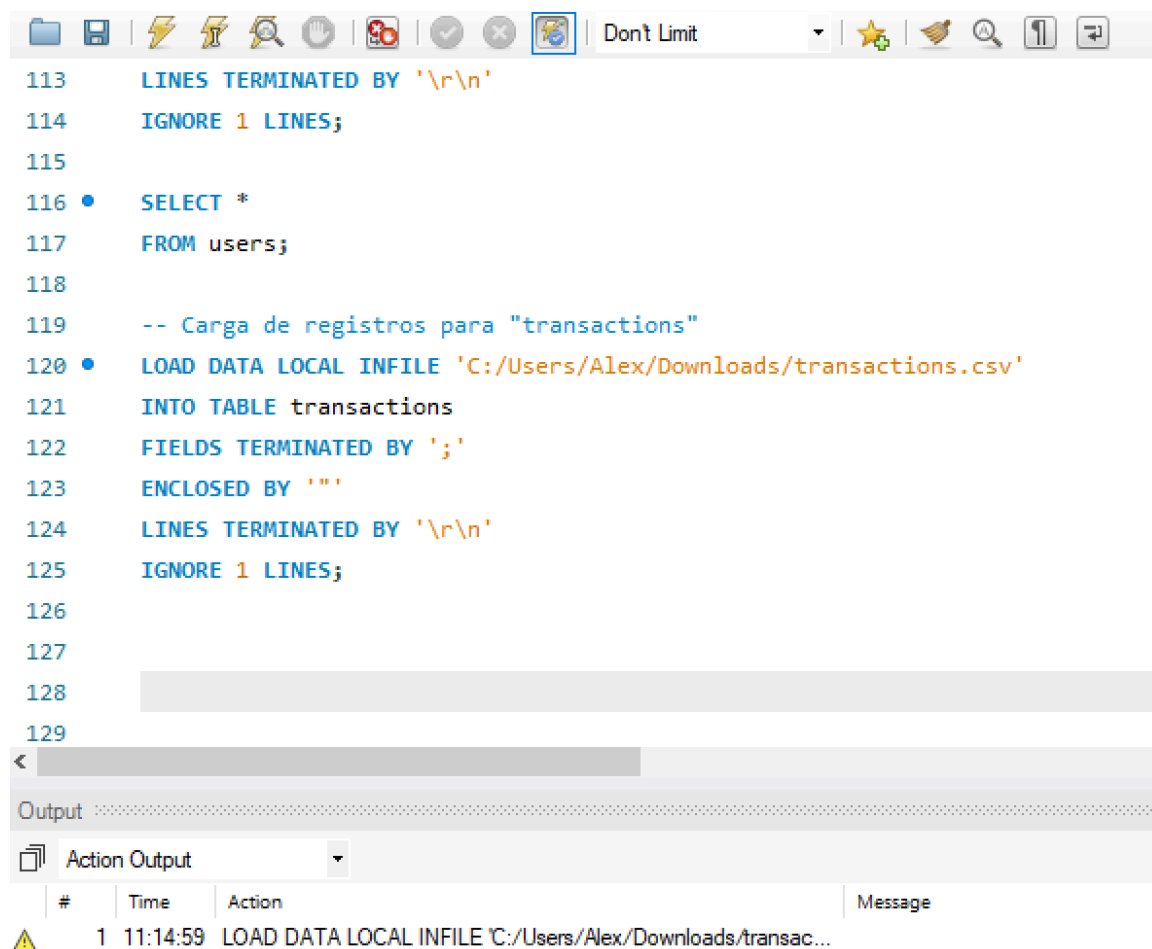
Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	10:59:28	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/user...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Wami...	0.016 sec
3	10:59:30	LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/user...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Wami...	0.031 sec
4	11:00:41	SELECT * FROM users	275 row(s) returned	0.000 sec / 0.016

Carga de registros para la tabla "transaction"

```
LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES;
```



```
113     LINES TERMINATED BY '\r\n'
114     IGNORE 1 LINES;
115
116 •   SELECT *
117     FROM users;
118
119     -- Carga de registros para "transactions"
120 •   LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/transactions.csv'
121     INTO TABLE transactions
122     FIELDS TERMINATED BY ';'
123     ENCLOSED BY '"'
124     LINES TERMINATED BY '\r\n'
125     IGNORE 1 LINES;
126
127
128
129
```

Output

Action Output

#	Time	Action	Message
1	11:14:59	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/transac...	

- Salta un warning, pero no indica donde el motivo del aviso. Sin embargo, al comprobar que la carga se ha realizado correctamente nos devuelve 587 registros, que coincide con los registros existentes en el archivo .csv

121 INTO TABLE transactions

122 FIELDS TERMINATED BY ','

123 ENCLOSED BY '"'

124 LINES TERMINATED BY '\r\n'

125 IGNORE 1 LINES;

126

127

128 • SELECT *

129 FROM transactions;

130

131

132

Result Grid

	id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat
▶	02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92	81.5
	0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.
	063FBA79-99EC-66FB-29F7-25726D1764A5	CcU-2987	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.
	0668296C-CDB9-A883-76BC-2E4C44F8C8AE	CcU-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265	-34.
	06CD9AA5-9B42-D684-DDDD-A5E394FEB999	CcU-2959	b-2346	2021-10-26 23:00:01	279.93	0	43, 31	92	33.7
	07A4ED48-21A2-7E87-6EB9-0DA007AD100E	CcU-2375	b-2386	2021-06-28 21:11:42	240.87	1	47, 33	275	38.6

transactions 20 x Apply

Output

Action Output

#	Time	Action	Message	Duration / F
1	11:14:59	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/transa...		0.047 sec
2	11:17:34	SELECT * FROM transactions	587 row(s) returned	0.000 sec /

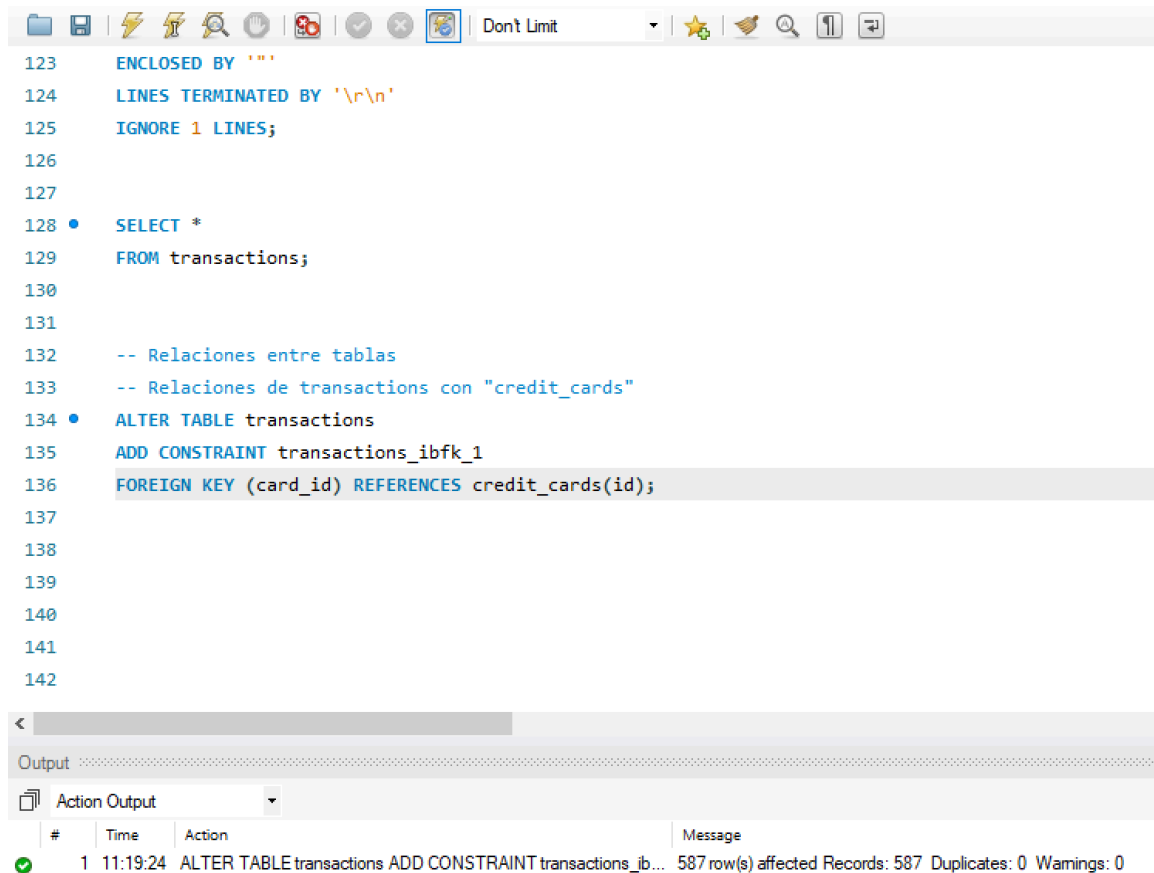
Creación de las relaciones entre tablas

La tabla "transactions" es la tabla de hechos, mientras que sus dimensiones son las tablas "companies", "users", y "credit_cards".

Procedemos a crear las relaciones entre la tabla de hechos y las de dimensiones para configurar un esquema de estrella

- Relación con "credit_cards"

```
ALTER TABLE transactions
ADD CONSTRAINT transactions_ibfk_1
FOREIGN KEY (card_id) REFERENCES credit_cards(id);
```



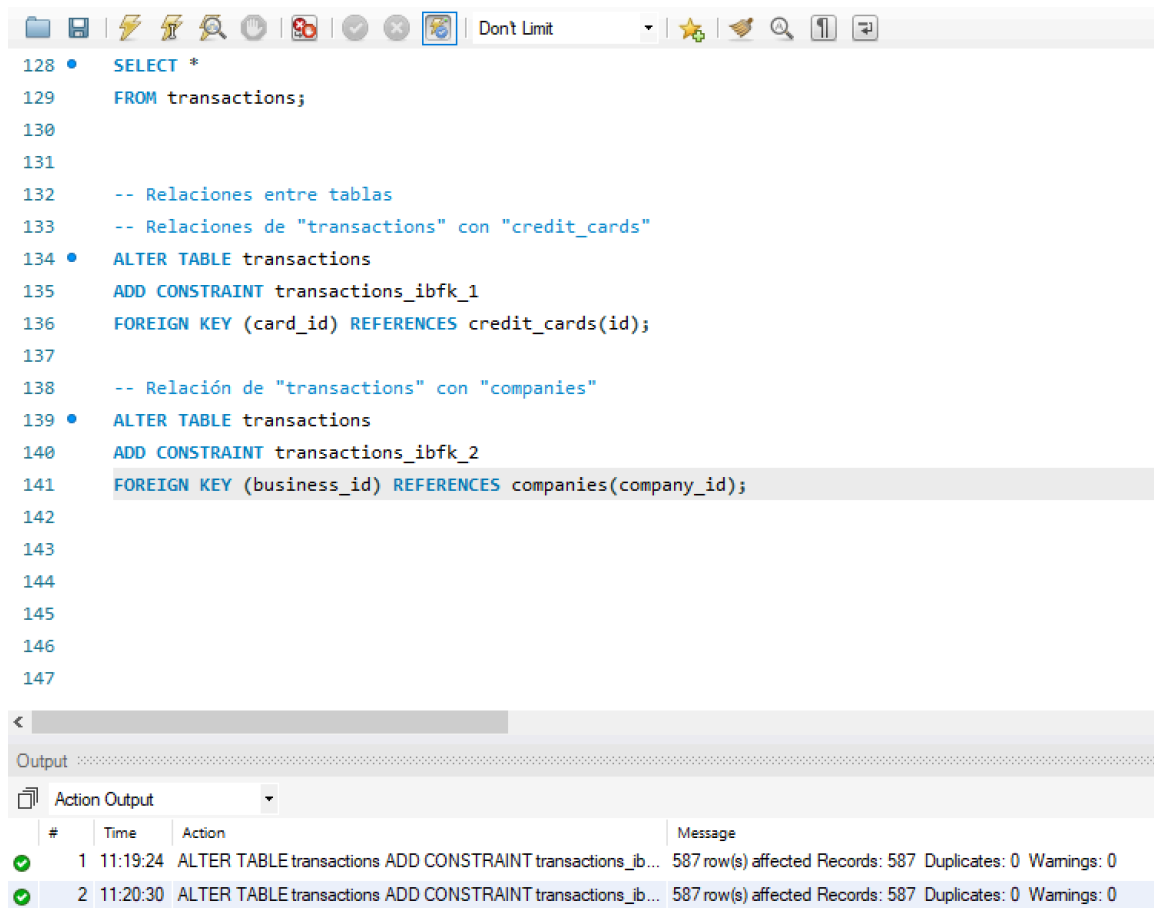
The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main area displays SQL code with line numbers 123 to 142. The code includes comments in Spanish and an SQL statement to add a foreign key constraint. Below the code, the 'Output' tab is active, showing a table with columns '#', 'Time', 'Action', and 'Message'. A single row indicates the successful execution of the ALTER TABLE statement, affecting 587 rows.

```
123 ENCLOSED BY ''''
124 LINES TERMINATED BY '\r\n'
125 IGNORE 1 LINES;
126
127
128 • SELECT *
129 FROM transactions;
130
131
132 -- Relaciones entre tablas
133 -- Relaciones de transactions con "credit_cards"
134 • ALTER TABLE transactions
135 ADD CONSTRAINT transactions_ibfk_1
136 FOREIGN KEY (card_id) REFERENCES credit_cards(id);
137
138
139
140
141
142
```

#	Time	Action	Message
✓ 1	11:19:24	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0

- Relación con "companies"

```
ALTER TABLE transactions
ADD CONSTRAINT transactions_ibfk_2
FOREIGN KEY (business_id) REFERENCES companies(company_id);
```



The screenshot shows a SQL IDE window with a toolbar at the top. The SQL editor contains the following code:

```

128 • SELECT *
129 FROM transactions;
130
131
132 -- Relaciones entre tablas
133 -- Relaciones de "transactions" con "credit_cards"
134 • ALTER TABLE transactions
135 ADD CONSTRAINT transactions_ibfk_1
136 FOREIGN KEY (card_id) REFERENCES credit_cards(id);
137
138 -- Relación de "transactions" con "companies"
139 • ALTER TABLE transactions
140 ADD CONSTRAINT transactions_ibfk_2
141 FOREIGN KEY (business_id) REFERENCES companies(company_id);
142
143
144
145
146
147

```

Below the editor is the 'Output' panel, which shows the 'Action Output' for the executed commands:

#	Time	Action	Message
✓ 1	11:19:24	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
✓ 2	11:20:30	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0

- Relación con "users"

```

ALTER TABLE transactions
ADD CONSTRAINT transactions_ibfk_3
FOREIGN KEY (user_id) REFERENCES users(id);

```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main area displays a series of SQL queries for the 'transactions' table, including comments in Spanish and three foreign key constraints. Below the queries, an 'Output' section shows the execution results for the three ALTER TABLE commands, each indicating that 587 rows were affected with no duplicates or warnings.

```

128 • SELECT *
129 FROM transactions;
130
131
132 -- Relaciones entre tablas
133 -- Relaciones de "transactions" con "credit_cards"
134 • ALTER TABLE transactions
135 ADD CONSTRAINT transactions_ibfk_1
136 FOREIGN KEY (card_id) REFERENCES credit_cards(id);
137
138 -- Relación de "transactions" con "companies"
139 • ALTER TABLE transactions
140 ADD CONSTRAINT transactions_ibfk_2
141 FOREIGN KEY (business_id) REFERENCES companies(company_id);
142
143 -- Relación de "transactions" con "users"
144 • ALTER TABLE transactions
145 ADD CONSTRAINT transactions_ibfk_3
146 FOREIGN KEY (user_id) REFERENCES users(id);
147
148

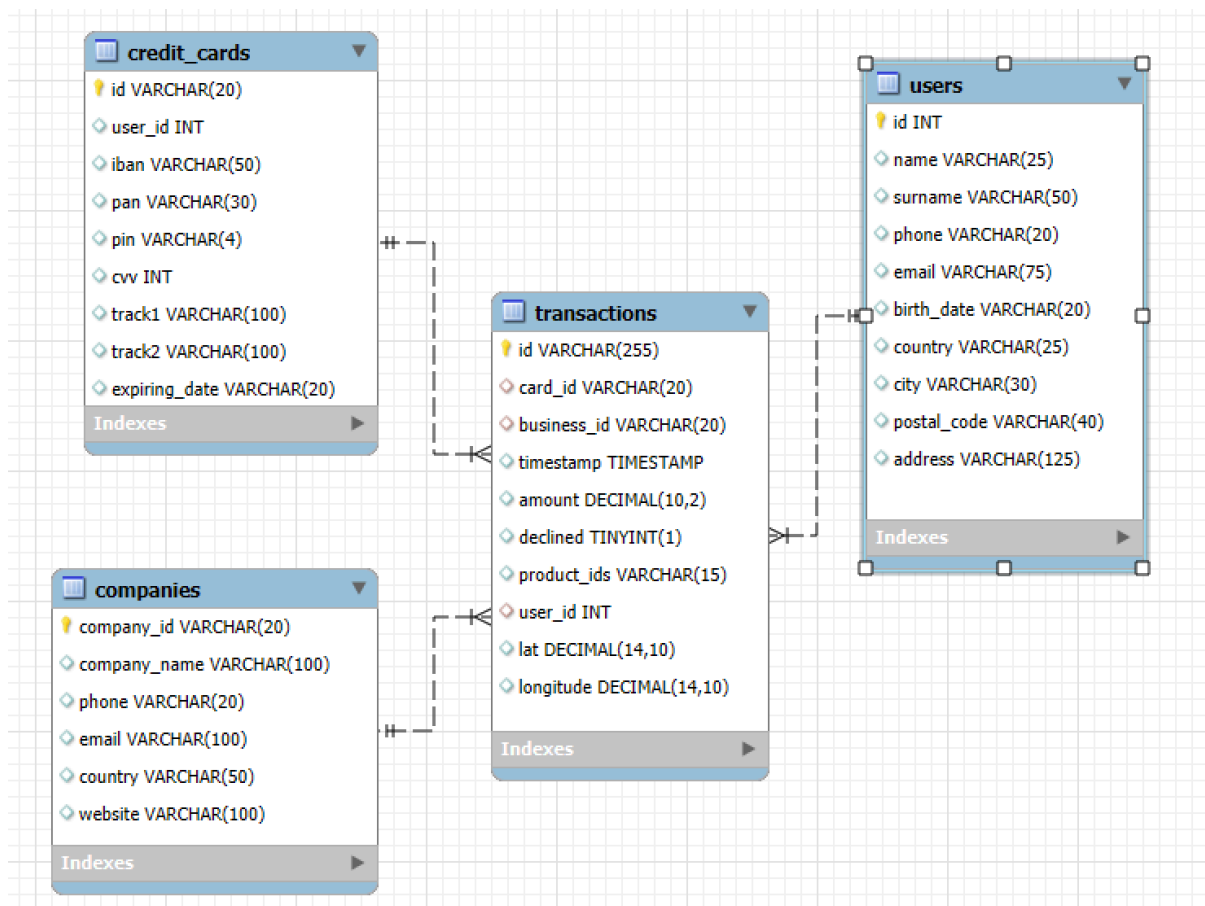
```

Output

Action Output

#	Time	Action	Message
✓ 1	11:19:24	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
✓ 2	11:20:30	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
✓ 3	11:21:38	ALTER TABLE transactions ADD CONSTRAINT transactions_ib...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0

Diagrama ER



Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.

Opció 1

- Creamos una subconsulta que nos devuelve el número de usuario de la tabla "transactions", basado en un filtraje combinado de HAVING y WHERE.
- En WHERE filtramos por todas aquellas transacciones que no han sido declinadas, y en HAVING filtramos por un conteo de todas las transacciones que son mayores a 30 y que cumplen el parámetro anterior. Agrupamos por número de usuario.
- En la consulta principal seleccionamos el id de usuario, el nombre y el apellido de la tabla "users". Usamos la función CONCAT en el nombre y

apellido para que se muestren de forma conjunta en una única tabla y le damos un alias para mayor claridad.

- Unimos la consulta y subconsulta con WHERE y IN a través del campo "id", que es PK de "users" y que se une a "transactions" a través de "user_id", su FK

```
SELECT
  id,
  CONCAT(name, " ", surname) AS full_name
FROM users
WHERE id IN (
  SELECT user_id
  FROM transactions
  WHERE declined = 0
  GROUP BY user_id
  HAVING COUNT(id) > 30
);
```

The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons and a 'Don't Limit' dropdown. Below the toolbar, a SQL query is written in a text editor. The query is as follows:

```

154 • SELECT
155     id,
156     CONCAT(name, " ", surname) AS full_name
157 FROM users
158 WHERE id IN (
159     SELECT user_id
160     FROM transactions
161     WHERE declined = 0
162     GROUP BY user_id
163     HAVING COUNT(id) > 30
164 );
165

```

Below the query editor, there's a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The grid displays the following data:

	id	full_name
▶	92	Lynn Riddle
	267	Ocean Nelson
	272	Hedwig Gilbert

Below the result grid, there's a 'Result 21' tab. Underneath it, there's an 'Output' section with a dropdown menu set to 'Action Output'. The output shows a single message:

#	Time	Action	Message
1	11:23:51	SELECT id, CONCAT(name, " ", surname) AS full_name FROM...	3 row(s) returned

Opción 2

- Nuestra tabla principal es "transactions" y configuramos la consulta principal para que nos devuelva el número de usuario y el conteo de todas las transacciones filtradas por aquellas que no hayan sido declinadas y por las que cuyo conteo sea superior a 30.
- Para obtener también el nombre de usuario y el apellido que solo se encuentran en la tabla "users", añadimos un par de subconsultas en SELECT seleccionando estos campos de esa tabla, igualando ambas tablas por su PK y FK.

- De esta forma, además de los nombres podremos obtener el número de transacciones en una columna.

```
SELECT
  user_id,
  (SELECT name FROM users WHERE users.id = t.user_id) AS name,
  (SELECT surname FROM users WHERE users.id = t.user_id) AS surname,
  COUNT(t.id) AS num_trans
FROM transactions AS t
WHERE declined = 0
GROUP BY user_id
HAVING num_trans > 30;
```


165
 166 -- Opción 2
 167 • SELECT
 168 user_id,
 169 (SELECT name FROM users WHERE users.id = t.user_id) AS name,
 170 (SELECT surname FROM users WHERE users.id = t.user_id) AS surname,
 171 COUNT(t.id) AS num_trans
 172 FROM transactions AS t
 173 WHERE declined = 0
 174 GROUP BY user_id
 175 HAVING num_trans > 30;
 176

Result Grid

user_id	name	surname	num_trans
92	Lynn	Riddle	39
267	Ocean	Nelson	39
272	Hedwig	Gilbert	38

Result 22 x

Output

Action Output

#	Time	Action	Message
1	11:23:51	SELECT id, CONCAT(name, " ", surname) AS full_name FROM...	3 row(s) returned
2	11:24:45	SELECT user_id, (SELECT name FROM users WHERE users.i...	3 row(s) returned

Entre ambas opciones, la opción 1 es más eficiente al usar una única subconsulta. Sin embargo, la opción 2 es más completa al mostrar también la cantidad exacta de transacciones realizadas de los tres clientes que coinciden con el criterio de haber realizado más de 30 transacciones.

Exercici 2

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

- Unimos la tablas "company", "transactions" y "credit_cards" mediante una INNER JOIN mediante sus PK y FK.
- Seleccionamos los campos "company_id" y "company_name" de la tabla "companies", además de calcular la media de "amount" de la tabla "transactions" y seleccionar el campo "iban" de "credit_cards".
- Filtramos por el identificador de Donec Ltd, ya que al tratarse de una PK la consulta será más eficiente, y por todas aquellas transacciones que no hayan sido declinadas.
- Finalmente, filtramos por el campo "iban" de la tabla "credit_card".

```
SELECT
  c.company_id,
  c.company_name,
  ROUND(AVG(t.amount),2) AS avg_amount,
  cc.iban
FROM companies AS c
JOIN transactions AS t
  ON c.company_id = t.business_id
JOIN credit_cards AS cc
  ON t.card_id = cc.id
WHERE c.company_id = "b-2242"
  AND declined = 0
GROUP BY cc.iban;
```

180

181 • **SELECT**

182 c.company_id,

183 c.company_name,

184 ROUND(AVG(t.amount),2) AS avg_amount,

185 cc.iban

186 **FROM** companies AS c

187 **JOIN** transactions AS t

188 **ON** c.company_id = t.business_id

189 **JOIN** credit_cards AS cc

190 **ON** t.card_id = cc.id

191 **WHERE** c.company_id = "b-2242"

192 **AND** declined = 0

193 **GROUP BY** cc.iban;

<

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

	company_id	company_name	avg_amount	iban
▶	b-2242	Donec Ltd	42.82	PT87806228135092429456346

Result 23 x

Output

Action Output ▼

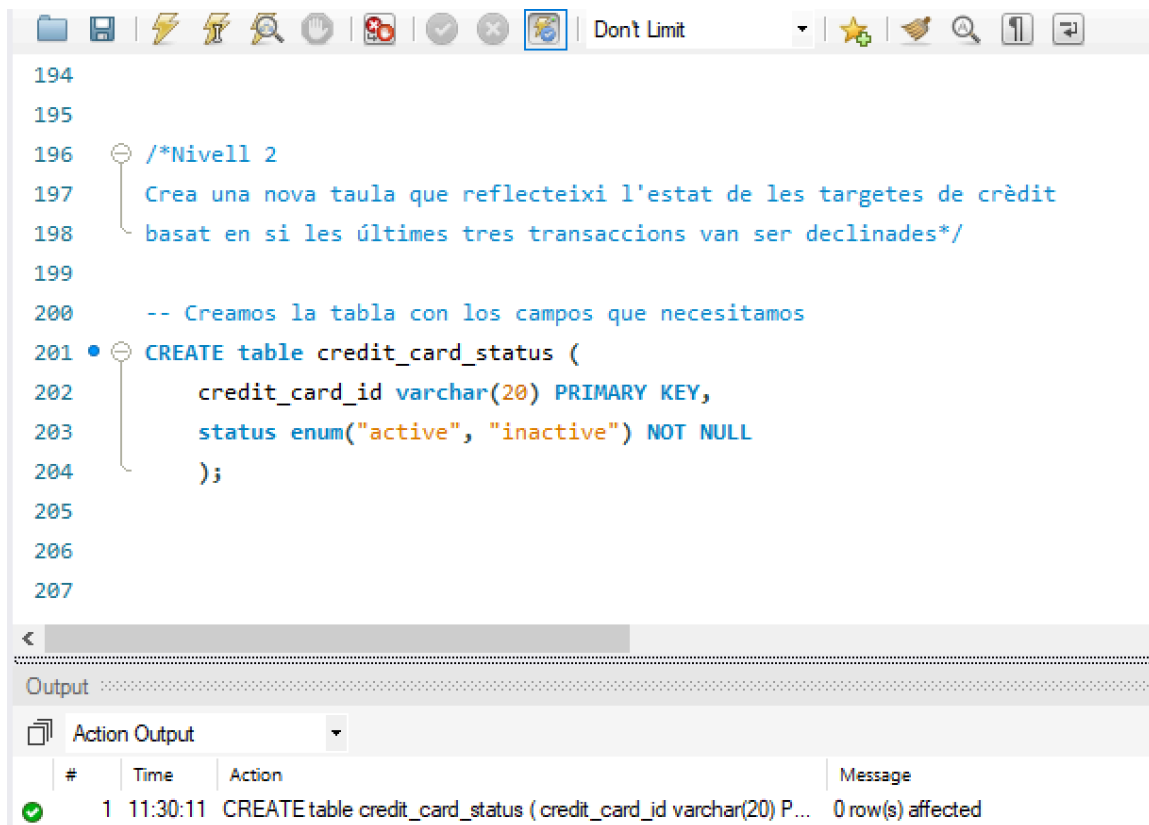
	#	Time	Action	Message
✓	2	11:24:45	SELECT user_id, (SELECT name FROM users WHERE user...	3 row(s) returned
✓	3	11:26:32	SELECT c.company_id, c.company_name, ROUND(AVG(t.a...	1 row(s) returned

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

1. Creamos la tabla en la cual insertaremos el identificador de tarjeta y su estado
 - a. Asignamos al campo un valor varchar(20), idéntico al de la tabla "credit_card", y le asignamos la categoría de PK.
 - b. Creamos el campo "status" como un enum o lista seleccionable, identificando dos posibles opciones: "active" o "inactive", dependiendo de la consulta que usaremos para obtener los datos que necesitamos para añadirlos a la tabla.
 - c. Vinculamos esta tabla con "credit_cards" a través del identificador de la tarjeta presente en ambas tablas

```
CREATE table credit_card_status (  
    credit_card_id varchar(20) PRIMARY KEY,  
    status enum("active", "inactive") NOT NULL  
);
```



The screenshot shows a database IDE with a toolbar at the top. The main editor displays SQL code with line numbers 194 to 207. The code includes a comment in Catalan and a SQL statement to create a table named `credit_card_status`. Below the editor is an 'Output' panel with a dropdown menu set to 'Action Output'. It contains a table with columns '#', 'Time', 'Action', and 'Message'. A single row shows a successful execution of the `CREATE table` statement at 11:30:11, with 0 rows affected.

```
194
195
196  /*Nivell 2
197   Crea una nova taula que reflecteixi l'estat de les targetes de crèdit
198   basat en si les últimes tres transaccions van ser declinades*/
199
200   -- Creamos la tabla con los campos que necesitamos
201  • CREATE table credit_card_status (
202      credit_card_id varchar(20) PRIMARY KEY,
203      status enum("active", "inactive") NOT NULL
204  );
205
206
207
```

Output

Action Output

#	Time	Action	Message
✓ 1	11:30:11	CREATE table credit_card_status (credit_card_id varchar(20) P...	0 row(s) affected

- Comprobamos que la tabla se haya creado correctamente

```
DESCRIBE credit_card_status;
```

200 -- Creamos la tabla con los campos que necesitamos

201 • CREATE table credit_card_status (

202 credit_card_id varchar(20) PRIMARY KEY,

203 status enum("active", "inactive") NOT NULL

204);

205

206 • DESCRIBE credit_card_status;

Result Grid

Field	Type	Null	Key	Default	Extra
credit_card_id	varchar(20)	NO	PRI	NULL	
status	enum('active','inactive')	NO		NULL	

Result 24 x

Output

Action Output

#	Time	Action	Message
✓ 1	11:30:11	CREATE table credit_card_status (credit_card_id varchar(20) ...	0 row(s) affected
✓ 2	11:31:08	DESCRIBE credit_card_status	2 row(s) returned

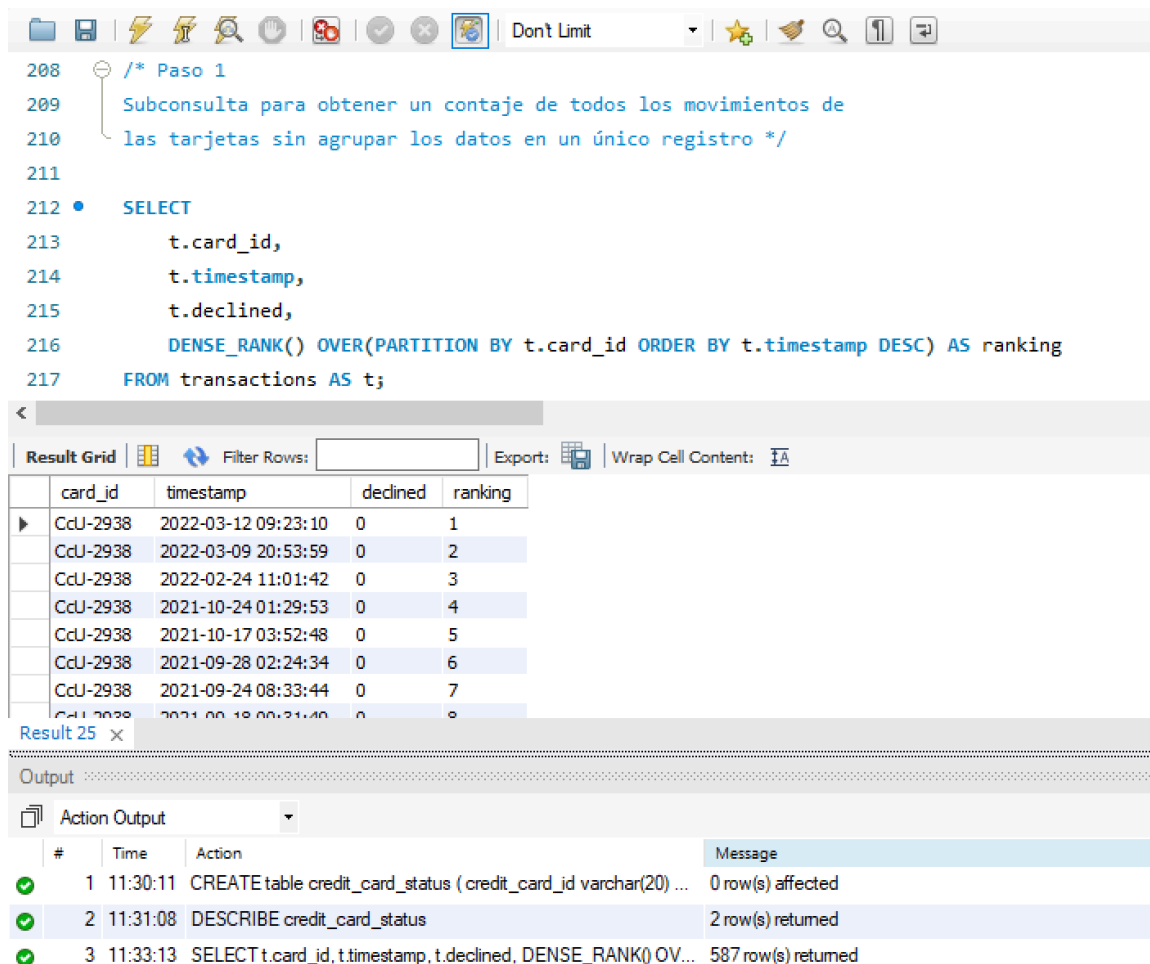
2. Creamos la consulta necesaria para obtener los resultados que necesitamos para insertarlos en la tabla creada

Subconsulta para obtener un conteo de todos los movimientos de las tarjetas sin agrupar los datos en un único registro

- Creamos una consulta para seleccionar los campos "card_id", "timestamp" y "declined" de la tabla "transactions".
- Usamos la window function DENSE_RANK() junto a PARTITION BY en el campo "card_id" para crear una nueva columna que agrupe las transacciones por tarjeta, asignándoles una numeración desde 1, y así seguir teniendo contexto de cada transacción en la que la tarjeta fue usada.
- Añadimos un ORDER BY en orden descendente para el campo "timestamp" para ordenar las transacciones de la más reciente a las más antigua.

- El número del ranking se reiniciará cuando cambie el identificador de la tarjeta.

```
SELECT
    t.card_id,
    t.timestamp,
    t.declined,
    DENSE_RANK() OVER(PARTITION BY t.card_id ORDER BY t.timestamp DESC) AS ranking
FROM transactions AS t;
```



The screenshot shows a SQL IDE interface. At the top, there's a toolbar with various icons. Below it, a SQL query is entered in a text area. The query is a SELECT statement that uses DENSE_RANK() to assign a ranking to transactions based on their timestamp, partitioned by card_id. The results are displayed in a table below the query. The table has four columns: card_id, timestamp, declined, and ranking. The data shows transactions for card_id 'CcU-2938' with timestamps ranging from 2021-09-24 to 2022-03-12, all with a declined status of 0. The ranking is assigned in descending order of timestamp, starting from 1 for the most recent transaction.

```
/* Paso 1
Subconsulta para obtener un contejo de todos los movimientos de
las tarjetas sin agrupar los datos en un único registro */

SELECT
    t.card_id,
    t.timestamp,
    t.declined,
    DENSE_RANK() OVER(PARTITION BY t.card_id ORDER BY t.timestamp DESC) AS ranking
FROM transactions AS t;
```

card_id	timestamp	declined	ranking
CcU-2938	2022-03-12 09:23:10	0	1
CcU-2938	2022-03-09 20:53:59	0	2
CcU-2938	2022-02-24 11:01:42	0	3
CcU-2938	2021-10-24 01:29:53	0	4
CcU-2938	2021-10-17 03:52:48	0	5
CcU-2938	2021-09-28 02:24:34	0	6
CcU-2938	2021-09-24 08:33:44	0	7
CcU-2938	2021-09-18 00:31:40	0	8

Result 25 x

Output

Action Output

#	Time	Action	Message
1	11:30:11	CREATE table credit_card_status (credit_card_id varchar(20) ...	0 row(s) affected
2	11:31:08	DESCRIBE credit_card_status	2 row(s) returned
3	11:33:13	SELECT t.card_id,t.timestamp,t.declined, DENSE_RANK() OV...	587 row(s) returned

Consulta para obtener los campos que necesitamos para insertarlos en la tabla

- Seleccionamos el campo "card_id" para obtener el identificador de tarjeta
- Mediante un CASE, indicamos que si esa tarjeta ha sido declinada 3 veces o más (valor 1 o TRUE) , será marcada como "inactive". Cualquier otro resultado, será marcado como "active".
- Filtramos por las 3 últimas transacciones realizadas, previamente ordenadas en orden descendiente en la subconsulta

```
SELECT
  card_id AS credit_card_id,
  CASE
    WHEN sum(declined) >= 3 THEN "inactive"
    ELSE "active"
  END AS status
FROM (SELECT
  t.card_id,
  t.timestamp,
  t.declined,
  DENSE_RANK() OVER(PARTITION BY t.card_id ORDER BY t.timestamp
  FROM transactions AS t
  ) AS rt
WHERE ranking <= 3 --Filtramos por las 3 últ. transacciones (las hemos orde
GROUP BY credit_card_id;
```


The screenshot shows a SQL IDE window with a query editor and a results pane. The query is as follows:

```

222 • SELECT
223     card_id AS credit_card_id,
224     CASE
225         WHEN sum(declined) >= 3 THEN "inactive"
226         ELSE "active"
227     END AS status
228 FROM (SELECT
229     t.card_id,
230     t.timestamp,
231     t.declined,
232     DENSE_RANK() OVER(PARTITION BY t.card_id ORDER BY t.timestamp DESC) AS ranking
233     FROM transactions AS t
234 ) AS rt
235 WHERE ranking <= 3 -- Filtramos por las 3 últ. transacciones (las hemos ordenado en DESC)
236 GROUP BY credit_card_id;

```

The results pane shows a table with the following data:

credit_card_id	status
CcU-2938	active
CcU-2945	active
CcU-2952	active
CcU-2959	active
CcU-2966	active

The output pane shows the following messages:

#	Time	Action	Message
3	11:33:13	SELECT t.card_id, t.timestamp, t.declined, DENSE_RANK() O...	587 row(s) returned
4	11:35:02	SELECT card_id AS credit_card_id, CASE WHEN sum(declin...	275 row(s) returned

El resultado de la consulta nos devuelve que las 275 tarjetas diferentes que están registradas están activas.

Inserción de la consulta en la tabla creada

```

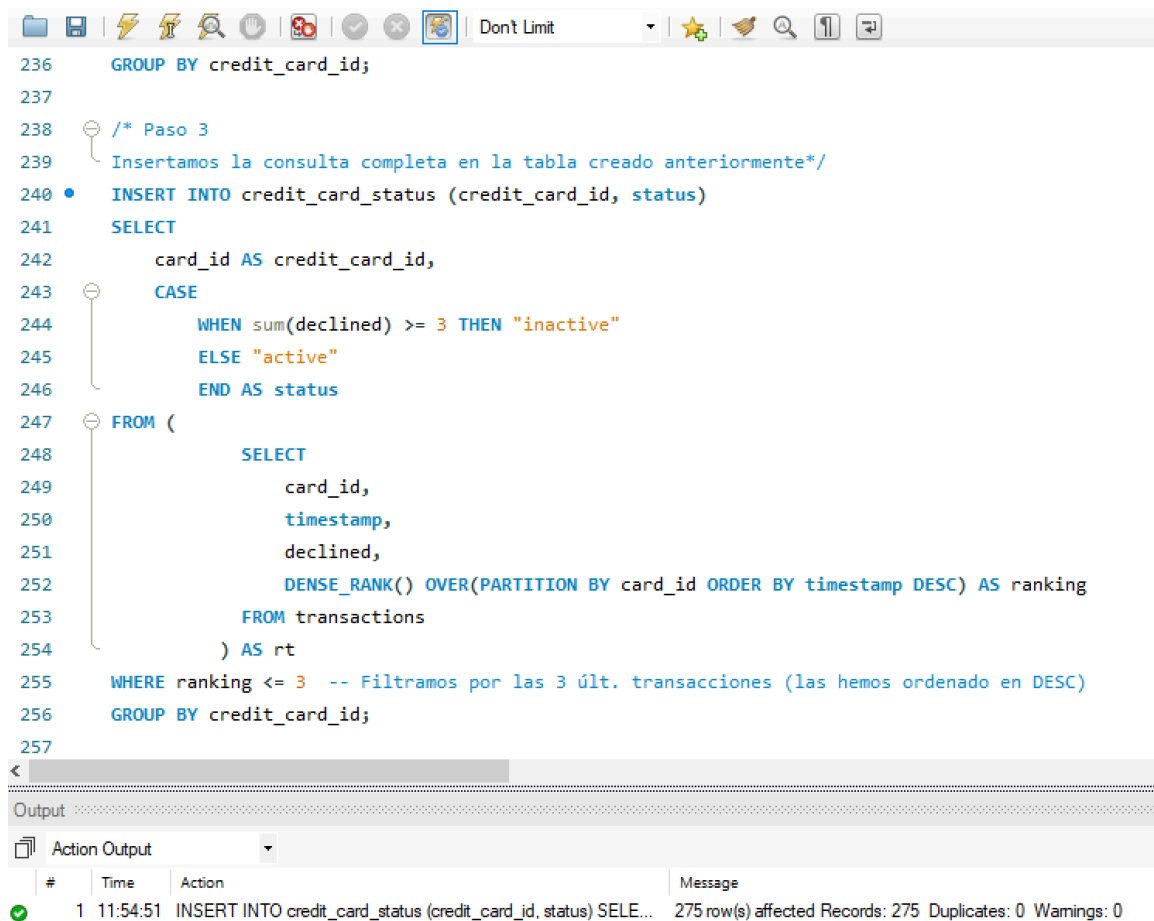
INSERT INTO credit_card_status (credit_card_id, status)
SELECT
    card_id AS credit_card_id,
    CASE
        WHEN sum(declined) < 3 THEN "inactive"
        ELSE "active"
    END AS status

```

```

FROM (
    SELECT
        card_id,
        timestamp,
        declined,
        DENSE_RANK() OVER(PARTITION BY card_id ORDER BY timestamp I
    FROM transactions
) AS rt
WHERE ranking <= 3 -- Filtramos por las 3 últ. transacciones (las hemos orde
GROUP BY credit_card_id;

```



The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The main editor displays a SQL script with line numbers 236 to 257. The script continues from the previous block, adding a CASE statement to determine the status based on the sum of declined transactions. A comment indicates this is 'Paso 3' and that the query is being inserted into a table. The output pane at the bottom shows a successful execution of the INSERT statement, affecting 275 rows with no duplicates or warnings.

```

236 GROUP BY credit_card_id;
237
238 /* Paso 3
239 Insertamos la consulta completa en la tabla creado anteriormente*/
240 • INSERT INTO credit_card_status (credit_card_id, status)
241 SELECT
242     card_id AS credit_card_id,
243     CASE
244         WHEN sum(declined) >= 3 THEN "inactive"
245         ELSE "active"
246     END AS status
247 FROM (
248     SELECT
249         card_id,
250         timestamp,
251         declined,
252         DENSE_RANK() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS ranking
253     FROM transactions
254 ) AS rt
255 WHERE ranking <= 3 -- Filtramos por las 3 últ. transacciones (las hemos ordenado en DESC)
256 GROUP BY credit_card_id;
257

```

Output

Action Output

#	Time	Action	Message
✓ 1	11:54:51	INSERT INTO credit_card_status (credit_card_id, status) SELE...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

- Comprobamos que los datos se han insertado correctamente

```
SELECT *
FROM credit_card_status;
```

The screenshot shows a SQL IDE interface. The top toolbar includes icons for file operations, execution, and settings. The main editor contains a SQL query with line numbers 251 to 260. The query is a CTE that ranks transactions by timestamp for each card ID, then selects all data from the credit_card_status table. Below the editor is a 'Result Grid' showing a table with columns 'credit_card_id' and 'status'. The table contains 11 rows of data, all with 'active' status. At the bottom, an 'Output' pane shows the execution log with a message: 'SELECT * FROM credit_card_status' and '275 row(s) returned'.

```
251         declined,
252         DENSE_RANK() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS ranking
253     FROM transactions
254 ) AS rt
255 WHERE ranking <= 3 -- Filtramos por las 3 últ. transacciones (las hemos ordenado en DESC)
256 GROUP BY credit_card_id;
257
258
259 • SELECT *
260 FROM credit_card_status;
```

credit_card_id	status
CcU-2938	active
CcU-2945	active
CcU-2952	active
CcU-2959	active
CcU-2966	active
CcU-2973	active
CcU-2980	active
CcU-2987	active
CcU-2994	active
CcU-3001	active

credit_card_status 31 x

Output

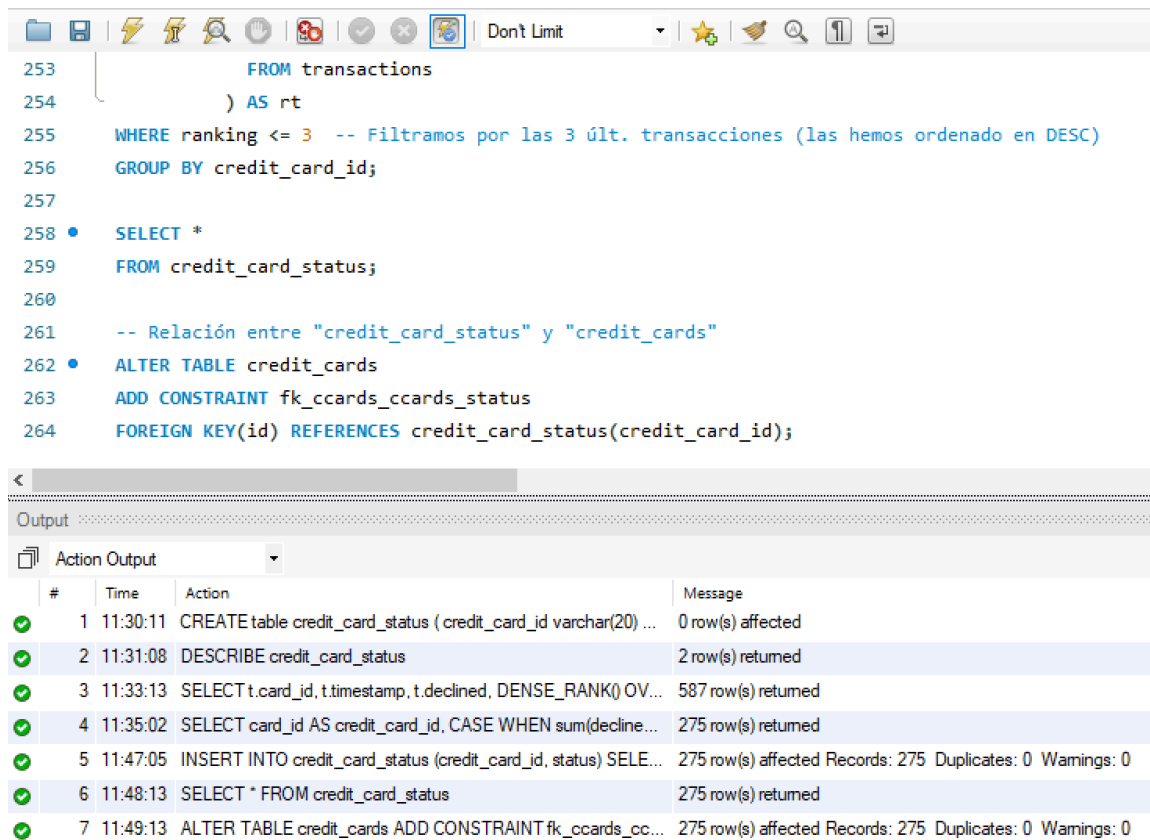
Action Output

#	Time	Action	Message
1	11:56:08	SELECT * FROM credit_card_status	275 row(s) returned

Relación entre "credit_card_status" y "credit_cards"

- Relacionamos ambas tablas a través de la PK "id" de la tabla "credit_cards", que hace de FK del PK "credit_card_id" de "credit_card_status"
- Se trata de una relación de 1 a N

```
ALTER TABLE credit_cards
ADD CONSTRAINT fk_ccards_ccards_status
FOREIGN KEY(id) REFERENCES credit_card_status(credit_card_id);
```



The screenshot shows a SQL IDE with a script editor and an output window. The script editor contains SQL code for creating a table, inserting data, and adding a foreign key constraint. The output window shows the execution results of these statements.

```
253     FROM transactions
254   ) AS rt
255   WHERE ranking <= 3 -- Filtramos por las 3 últ. transacciones (las hemos ordenado en DESC)
256   GROUP BY credit_card_id;
257
258 • SELECT *
259   FROM credit_card_status;
260
261   -- Relación entre "credit_card_status" y "credit_cards"
262 • ALTER TABLE credit_cards
263   ADD CONSTRAINT fk_ccards_ccards_status
264   FOREIGN KEY(id) REFERENCES credit_card_status(credit_card_id);
```

Output

Action Output

#	Time	Action	Message
✓ 1	11:30:11	CREATE table credit_card_status (credit_card_id varchar(20) ...	0 row(s) affected
✓ 2	11:31:08	DESCRIBE credit_card_status	2 row(s) returned
✓ 3	11:33:13	SELECT t.card_id, t.timestamp, t.declined, DENSE_RANK() OV...	587 row(s) returned
✓ 4	11:35:02	SELECT card_id AS credit_card_id, CASE WHEN sum(decline...	275 row(s) returned
✓ 5	11:47:05	INSERT INTO credit_card_status (credit_card_id, status) SELE...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
✓ 6	11:48:13	SELECT * FROM credit_card_status	275 row(s) returned
✓ 7	11:49:13	ALTER TABLE credit_cards ADD CONSTRAINT fk_ccards_cc...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

Exercici 1

Quantes targetes estan actives?

```
SELECT COUNT(*) AS active_cards
FROM credit_card_status
WHERE status = "active";
```

The screenshot shows a database management tool interface. The top toolbar includes icons for file operations, execution, and search. The main area displays SQL code with line numbers 263 to 275. The code includes an ALTER TABLE statement to add a foreign key constraint and a SELECT statement to count active cards. Below the code, a 'Result Grid' shows a single row with the value '275' for the column 'active_cards'. At the bottom, an 'Output' pane shows the execution log with a message: 'SELECT COUNT(*) AS active_cards FROM credit_card_status ... 1 row(s) returned'.

```

263
264 • ALTER TABLE credit_cards
265     ADD CONSTRAINT fk_ccards_ccards_status
266     FOREIGN KEY(id) REFERENCES credit_card_status(credit_card_id);
267
268
269 /* Exercici 1
270    Quantes targetes estàn actives? */
271 • SELECT COUNT(*) AS active_cards
272     FROM credit_card_status
273     WHERE status = "active";
274
275

```

active_cards
275

Result 32 x

Output

Action Output

#	Time	Action	Message
1	11:56:59	SELECT COUNT(*) AS active_cards FROM credit_card_status ...	1 row(s) returned

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.

- Creamos la tabla "products"
 - Asignamos el valor de Primary Key al campo "id"

```

CREATE TABLE IF NOT EXISTS products (
    id int,
    product_name varchar(50),
    price varchar(10),
    colour varchar(10),
    weight decimal(6,2),
    warehouse_id varchar(10),
    PRIMARY KEY (id)
);

```

The screenshot shows a SQL IDE window with a toolbar at the top. The main area contains the following SQL code:

```

271 • SELECT COUNT(*) AS active_cards
272 FROM credit_card_status
273 WHERE status = "active";
274
275
276 /* Nivell 3
277 Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la
278 base de dades creada, tenint en compte que des de transaction tens product_ids. */
279
280 -- Creamos la tabla "products"
281 • CREATE TABLE IF NOT EXISTS products (
282     id int,
283     product_name varchar(50),
284     price varchar(10),
285     colour varchar(10),
286     weight decimal(6,2),
287     warehouse_id varchar(10),
288     PRIMARY KEY (id)
289 );
290
291

```

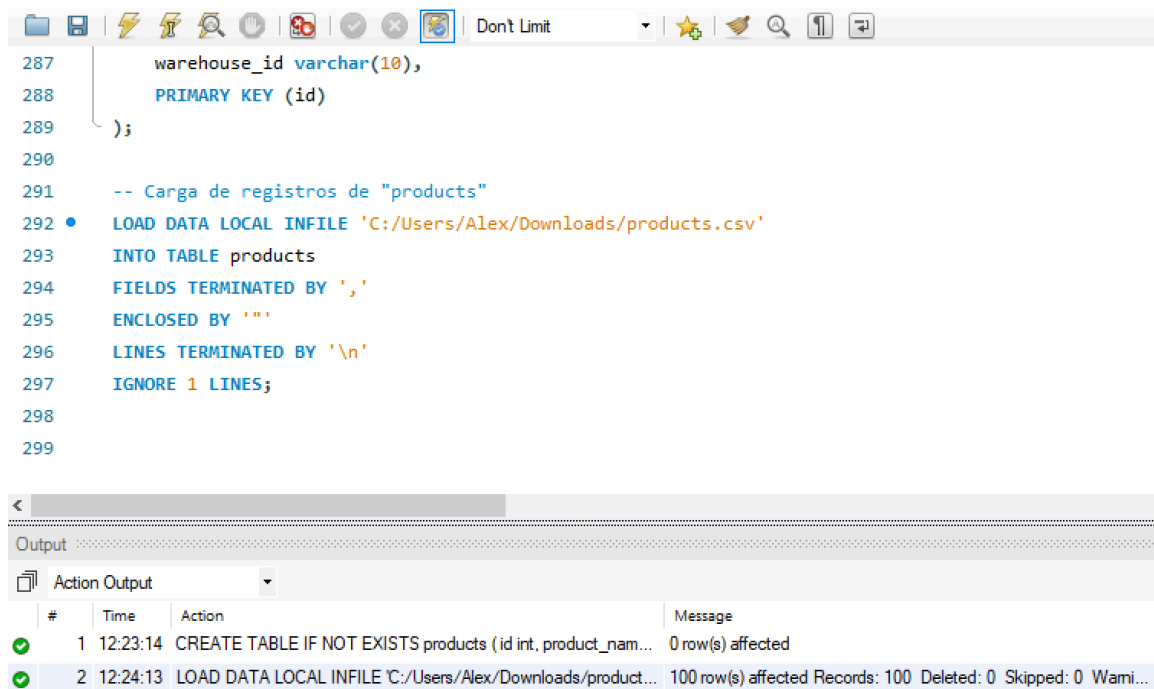
The screenshot shows the Output window of the SQL IDE. It displays the execution of the CREATE TABLE statement. The table has columns: #, Time, Action, and Message.

#	Time	Action	Message
1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_name...	0 row(s) affected

Carga de registros para la tabla "products"

- Ignoramos la primera fila del .csv al tratarse del encabezado de las columnas

```
LOAD DATA LOCAL INFILE "C:/Users/Alex/Downloads/products.csv"
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;
```



The screenshot shows a SQL IDE window with a script editor and an output pane. The script editor contains the following SQL code:

```
287     warehouse_id varchar(10),
288     PRIMARY KEY (id)
289 );
290
291 -- Carga de registros de "products"
292 • LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/products.csv'
293 INTO TABLE products
294 FIELDS TERMINATED BY ','
295 ENCLOSED BY '"'
296 LINES TERMINATED BY '\n'
297 IGNORE 1 LINES;
298
299
```

The output pane shows the results of the script execution:

#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_nam...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/product...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Wami...

- Comprobamos que los datos se han cargado correctamente

295 ENCLOSED BY ''''

296 LINES TERMINATED BY '\n'

297 IGNORE 1 LINES;

298

299 • SELECT *

300 FROM products;

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
9	Winterfell	\$76.40	#b5b5b5	1.20	WH--4
10	Karstark Dorne	\$119.52	#f4f4f4	2.40	WH--5
11	Karstark Dorne	\$49.70	#141414	2.70	WH--6
12	duel Direwolf	\$181.60	#a8a8a8	2.10	WH--7

products 33 x

Output

#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_na...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/produ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 War...
✓ 3	12:25:06	SELECT * FROM products	100 row(s) returned

Normalización de la tabla

El campo "price" es un varchar y queremos transformarlo a decimal.

- Para ello, primero eliminamos el simbolo del dolar del campo con la función TRIM, y añadimos LEADING porque el símbolo del dolar está al principio de la cadena (si estuviera al final usaríamos TRAILING)

```
UPDATE products
SET price = TRIM(LEADING '$' FROM price);
```



```

302 -- Transformamos el campo "price" a decimal
303 • UPDATE products
304 SET price = TRIM(LEADING '$' FROM price);

```

Output			
Action Output			
#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_na...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/produ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 War...
✓ 3	12:25:06	SELECT * FROM products	100 row(s) returned
✓ 4	12:26:09	UPDATE products SET price = TRIM(LEADING '\$' FROM price)	100 row(s) affected Rows matched: 100 Changed: 100 Wami...

- Comprobamos que el cambio se ha aplicado correctamente

303 • **UPDATE** products
 304 **SET** price = TRIM(LEADING '\$' FROM price);
 305
 306 • **SELECT** *
 307 **FROM** products;

Result Grid | Filter Rows: | Edit: | Export/Import:

	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
	9	Winterfell	76.40	#b5b5b5	1.20	WH--4

products 34 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_na...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/produ...	100 row(s) affected R
✓ 3	12:25:06	SELECT * FROM products	100 row(s) returned
✓ 4	12:26:09	UPDATE products SET price = TRIM(LEADING '\$' FROM price)	100 row(s) affected R
✓ 5	12:27:04	SELECT * FROM products	100 row(s) returned

- A continuación, le asignamos un valor numérico decimal de 8 dígitos de los cuales 2 son decimales.

```
ALTER TABLE products
MODIFY COLUMN price dec(8,2);
```

```

309      -- Lo convertimos a decimal
310 •    ALTER TABLE products
311      MODIFY COLUMN price dec(8,2);

```

Output

#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_na...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/produ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 War..
✓ 3	12:25:06	SELECT * FROM products	100 row(s) returned
✓ 4	12:26:09	UPDATE products SET price = TRIM(LEADING '\$' FROM price)	100 row(s) affected Rows matched: 100 Changed: 100 Wami..
✓ 5	12:27:04	SELECT * FROM products	100 row(s) returned
✓ 6	12:28:11	ALTER TABLE products MODIFY COLUMN price dec(8,2)	100 row(s) affected Records: 100 Duplicates: 0 Warnings: 0

- Comprobamos que el cambio se ha aplicado correctamente

```
DESCRIBE products;
```

313 • `DESCRIBE products;`

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
product_name	varchar(50)	YES		NULL	
price	decimal(8,2)	YES		NULL	
colour	varchar(10)	YES		NULL	
weight	decimal(6,2)	YES		NULL	
warehouse_id	varchar(10)	YES		NULL	

Result 35 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:23:14	CREATE TABLE IF NOT EXISTS products (id int, product_na...	0 row(s) affected
✓ 2	12:24:13	LOAD DATA LOCAL INFILE 'C:/Users/Alex/Downloads/produ...	100 row(s) affected
✓ 3	12:25:06	SELECT * FROM products	100 row(s) returned
✓ 4	12:26:09	UPDATE products SET price = TRIM(LEADING '\$' FROM price)	100 row(s) affected
✓ 5	12:27:04	SELECT * FROM products	100 row(s) returned
✓ 6	12:28:11	ALTER TABLE products MODIFY COLUMN price dec(8,2)	100 row(s) affected
✓ 7	12:29:05	DESCRIBE products	6 row(s) returned

Creación de una tabla intermedia entre “transactions” y “products”

La tabla “transactions” tiene un campo llamado “product_ids” con más de un valor en el campo, recogiendo que una misma transacción puede haber servido para comprar más de un producto.

“Product_ids” es un campo mal normalizado porque almacena varios valores almacenados separados por comas. Este diseño de la tabla no cumple la primera norma formal (1NF), en la cual debemos asegurarnos que cada columna contiene valores atómicos o indivisibles.

En este caso, es aconsejable crear una tabla intermedia en la cual pueda recogerse que una misma transacción puede contener varios productos (un producto por registro) y eliminar el campo "product_ids" de "transactions". Esta tabla intermedia también nos ayudará a establecer la correcta relación entre tablas.

Normalización del campo "product_ids" de "transactions" para cumplir la 1NF

- Los distintos valores se encuentran separados por comas y en algunos registros se trata de 2 valores, en otros de 3 y en otros de 4.
- Vamos a suponer que sabemos que el campo tiene varios valores, pero no sabemos el máximo de valores presentes en un campo entre todos ellos. Supongamos que contiene hasta un máximo de 7 valores.

Creación de la tabla intermedia "orders"

- Para poder repetir o bien el id de transacción o el identificador de producto en diferentes registros, necesitamos que la combinación de ambos sea la Primary Key de la tabla (Primary Key compuesta).

```
CREATE TABLE IF NOT EXISTS orders (  
  transaction_id VARCHAR(100) NOT NULL,  
  product_id INT NOT NULL,  
  PRIMARY KEY (transaction_id, product_id)  
);
```

```

315      -- Creación de la tabla intermedia "orders"
316 • CREATE TABLE IF NOT EXISTS orders (
317     transaction_id VARCHAR(100) NOT NULL,
318     product_id INT NOT NULL,
319     PRIMARY KEY (transaction_id, product_id)
320 );

```

Output			
Action Output			
#	Time	Action	Message
✓ 1	12:29:58	CREATE TABLE IF NOT EXISTS orders (transaction_id VARC...	0 row(s) affected

- Comprobamos que la tabla se haya creado correctamente

```

322 • DESCRIBE orders;

```

<

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	transaction_id	varchar(100)	NO	PRI	NULL	
	product_id	int	NO	PRI	NULL	

Result 36

×

Output

Action Output

	#	Time	Action	Message
✓	1	12:29:58	CREATE TABLE IF NOT EXISTS orders (transaction_id VARC...	0 row(s) affected
✓	2	12:30:42	DESCRIBE orders	2 row(s) returned

Normalización del campo multivalor "product_ids"

- Después de probar varias opciones para repartir los múltiples valores que contiene "product_ids" en varios campos para cumplir con la norma 1NF sin éxito, me quedo con la opción de unir las tablas "transactions" y "products" mediante un JOIN usando FIND_IN_SET.
 - Seleccionamos el campo identificador de las transacciones de la tabla "transactions", y el campo identificador de los productos de la tabla "products", la tabla de dimensiones en la que tenemos un identificador de producto por registro.
 - Hacemos un INNER JOIN de ambas tablas y las unimos mediante la función para cadenas de texto FIND IN SET, que buscará si los valores de "products.id", su primera variable, se encuentran en "transactions.product_ids", su segunda variable, y de forma interna nos devolverá su posición en el último campo, devolviéndonos un 0 si no hay una coincidencia entre valores.
 - Dentro de FIND IN SET usamos la función REPLACE anidando en la segunda variable mencionada. Con esto conseguimos eliminar todos los espacios vacíos dentro de ese campo.
 - Finalmente, usamos el operador lógico ">" para obtener las coincidencias entre campos, que corresponderán a los valores mayores que 0.

```
SELECT
  t.id as transaction_id,
  p.id AS product_id
FROM transactions as t
JOIN products as p
  ON FIND_IN_SET(p.id, REPLACE(t.product_ids, " ", "")) > 0;
```

324 -- Normalización del campo multivalor "product_ids"

325 • **SELECT**

326 t.id as transaction_id,

327 p.id AS product_id

328 **FROM** transactions as t

329 **JOIN** products as p

330 **ON** FIND_IN_SET(p.id, REPLACE(t.product_ids, " ", "")) > 0;

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

transaction_id	product_id
0A476ED9-0C13-1962-F87B-D3563924B539	11
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	41
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	29
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	19
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	3
0C7C3A33-9947-3BC1-846D-7BE3D0D17598	89
0C7C3A33-9947-3BC1-846D-7BE3D0D17598	31
0CE957A6-CCAA-2B7A-6839-8A4B1B324853	83
0CE957A6-CCAA-2B7A-6839-8A4B1B324853	73
0CE957A6-CCAA-2B7A-6839-8A4B1B324853	61
0CE957A6-CCAA-2B7A-6839-8A4B1B324853	43
0DD2E608-5C9E-D1B3-4999-B99F43AD735A	47
0DD2E608-5C9E-D1B3-4999-B99F43AD735A	17

Result 37 x

Output

Action Output

#	Time	Action	Message
1	12:29:58	CREATE TABLE IF NOT EXISTS orders (transaction_id VARC...	0 row(s) affected
2	12:30:42	DESCRIBE orders	2 row(s) returned
3	12:31:29	SELECT t.id as transaction_id, p.id AS product_id FROM trans...	1457 row(s) returned

- Insertamos los resultados en la tabla intermedia "orders" que hemos creado anteriormente mediante el comando INSERT INTO.

```
INSERT INTO orders (transaction_id, product_id)
SELECT
    t.id as transaction_id,
```



```

p.id AS product_id
FROM transactions AS t
JOIN products AS p
ON FIND_IN_SET(p.id, REPLACE(t.product_ids, " ", "")) > 0;

```

```

332 -- Insertamos los registros en la tabla creada anteriormente
333 • INSERT INTO orders (transaction_id, product_id)
334 SELECT
335     t.id as transaction_id,
336     p.id AS product_id
337 FROM transactions AS t
338 JOIN products AS p
339     ON FIND_IN_SET(p.id, REPLACE(t.product_ids, " ", "")) > 0;

```

Output

Action Output

#	Time	Action	Message
✓ 2	12:30:42	DESCRIBE orders	2 row(s) returned
✓ 3	12:31:29	SELECT t.id as transaction_id, p.id AS product_id FROM tran...	1457 row(s) returned
✓ 4	12:32:45	INSERT INTO orders (transaction_id, product_id) SELECT t.i...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Relación entre la tabla intermedia "orders" y el resto de tablas

La tabla "orders" tendrá dos relaciones, una hacia "transactions" y otra hacia "products", actuando como tabla intermedia que servirá como puente de unión entre ambas tablas.

- Relación entre "orders" y "products"

```

-- Relación entre "orders" y "products"
ALTER TABLE orders
ADD CONSTRAINT fk_orders_products
FOREIGN KEY (product_id) REFERENCES products(id);

```

```
-- Relación entre "orders" y "products"
```

- `ALTER TABLE orders`
`ADD CONSTRAINT fk_orders_products`
`FOREIGN KEY (product_id) REFERENCES products(id);`

ction Output

	Time	Action	Message
3	12:31:29	SELECT t.id as transaction_id, p.id AS product_id FROM tran...	1457 row(s) returned
4	12:32:45	INSERT INTO orders (transaction_id, product_id) SELECT t.i...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
5	12:33:56	ALTER TABLE orders ADD CONSTRAINT fk_orders_product...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

- Relación entre "orders" y "transactions"

```
-- Relación entre "orders" y "transactions"
ALTER TABLE orders
ADD CONSTRAINT fk_orders_transactions
FOREIGN KEY (transaction_id) REFERENCES transactions(id);
```

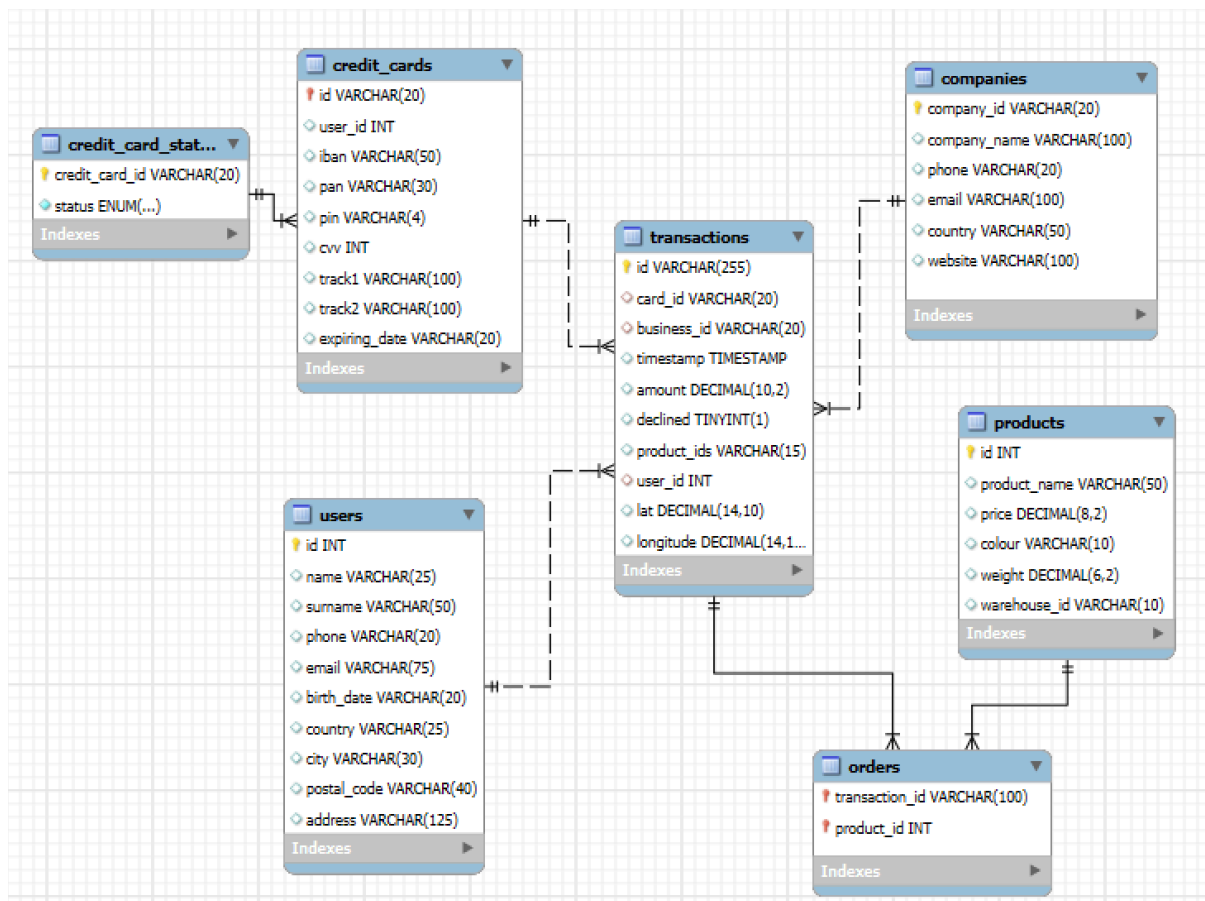
```
-- Relación entre "orders" y "products"
```

- `ALTER TABLE orders`
`ADD CONSTRAINT fk_orders_products`
`FOREIGN KEY (product_id) REFERENCES products(id);`

ction Output

	Time	Action	Message
3	12:31:29	SELECT t.id as transaction_id, p.id AS product_id FROM tran...	1457 row(s) returned
4	12:32:45	INSERT INTO orders (transaction_id, product_id) SELECT t.i...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
5	12:33:56	ALTER TABLE orders ADD CONSTRAINT fk_orders_product...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Diagrama ER



A diferencia de las relaciones entre las otras tablas que tienen una línea punteada uniéndolas, la relación entre "products", "orders" y transactions" muestra una línea sólida entre tablas.

Esto es como consecuencia de la relación identificada que une a las tres tablas, donde la tabla intermedia no puede ser correctamente identificada sin su tabla madre, teniendo además una Primary Key compuesta hecha de las dos Primary Key de las dos tablas madre que vincula.

Exercici 1

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

- Seleccionamos el campo "product_id" de la tabla y realizamos un conteo sobre el mismo campo para obtener el identificador de producto y el número de veces que aparece en la tabla "orders".
- Para filtrar por aquellas transacciones que no han sido declinadas en WHERE, hacemos un JOIN con transactions para recuperar el campo "declined".
- Finalmente, agrupamos los resultados por identificador de producto y ordenamos los resultados por unidades vendidas de mayor a menor.

```
SELECT
  o.product_id,
  COUNT(o.product_id) AS units_sold
FROM orders AS o
JOIN transactions as t
  ON t.id = o.transaction_id
WHERE t.declined = 0
GROUP BY o.product_id
ORDER BY units_sold DESC;
```

Don't Limit

```

354 • SELECT
355     o.product_id,
356     COUNT(o.product_id) AS units_sold
357 FROM orders AS o
358 JOIN transactions as t
359     ON t.id = o.transaction_id
360 WHERE t.declined = 0
361 GROUP BY o.product_id
362 ORDER BY units_sold DESC;
363

```

Result Grid
Filter Rows:
Export:
Wrap Cell Content:

	product_id	units_sold
▶	23	60
	67	59
	2	56
	43	54
	17	54
	97	53
	79	52
	1	51
	12	51

Result 38

Output

Action Output

#	Time	Action	Message
✓ 1	12:41:21	SELECT o.product_id, COUNT(o.product_id) AS units_sold F...	26 row(s) returned