

# 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:*

## Descripción de la base de datos:

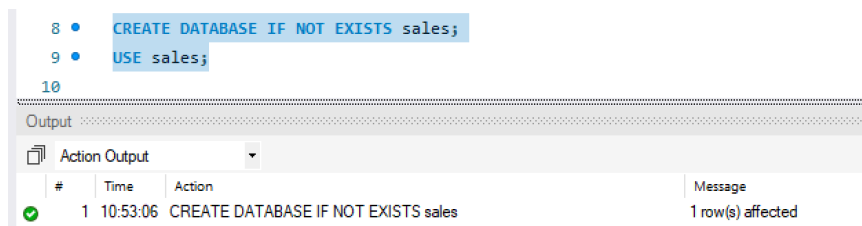
Inicialmente tenemos 7 tablas, pero 3 de ellas se pueden unificar en una, ya que se trata de 3 tablas con las mismas columnas de usuarios de nacionalidades de Canadá, USA y UK.

Tenemos por lo tanto 4 tablas: companies, users, credit\_card y transactions.

Se trata de una base de datos relacional estructurada en un esquema de modelo dimensional de estrella con la tabla 'transactions' como la tabla de hechos y las tablas 'users', 'credit\_card', 'products' y 'companies' como las tablas de dimensiones que se relacionan con la de hechos 1 a N. Existe también una relación entre las tabla users y credit\_cards 1 a N, que consideramos no es necesaria y procederemos a eliminar. Haremos también las modificaciones necesarias para optimizar y ordenar la base de datos: cambiar los tipos de birth\_date y expiring\_date de VARCHAR a DATE.

## Creemos la base de datos a la que hemos llamado sales:

CREATE DATABASE IF NOT EXISTS sales;



## Creemos las tablas:

```
CREATE TABLE IF NOT EXISTS credit_cards (  
    id VARCHAR(20) PRIMARY KEY,  
    user_id INT NOT NULL,  
    iban VARCHAR(50) UNIQUE NOT NULL,  
    pan VARCHAR(50) UNIQUE NOT NULL,  
    pin VARCHAR(50) NOT NULL,  
    cvv INT UNIQUE NOT NULL,  
    track1 VARCHAR(50),  
    track2 VARCHAR(50),  
    expiring_date VARCHAR(10) NOT NULL);
```

```

10 DROP TABLE credit_cards;
11 CREATE TABLE IF NOT EXISTS credit_cards (
12     id VARCHAR(20) PRIMARY KEY,
13     user_id INT NOT NULL,
14     iban VARCHAR(50) UNIQUE NOT NULL,
15     pan VARCHAR(50) UNIQUE NOT NULL,
16     pin VARCHAR(50) NOT NULL,
17     cvv INT NOT NULL,
18     track1 VARCHAR(50),
19     track2 VARCHAR(50),
20     expiring_date VARCHAR(10) NOT NULL);

```

Output

#	Time	Action	Message
✓ 13	11:59:23	CREATE TABLE IF NOT EXISTS credit_cards (id VARCHA...	0 row(s) affected

CREATE TABLE IF NOT EXISTS **companies** (  
 company\_id VARCHAR(20) PRIMARY KEY,  
 company\_name VARCHAR(50) NOT NULL,  
 phone VARCHAR(20) UNIQUE,  
 email VARCHAR(50),  
 country VARCHAR(20),  
 website VARCHAR(100));

```

22 CREATE TABLE IF NOT EXISTS companies (
23     company_id VARCHAR(20) PRIMARY KEY,
24     company_name VARCHAR(50) NOT NULL,
25     phone VARCHAR(20) UNIQUE,
26     email VARCHAR(50),
27     country VARCHAR(20),
28     website VARCHAR(100));
29

```

Output

#	Time	Action	Message
✓ 3	11:18:33	CREATE TABLE IF NOT EXISTS credit_cards (id VARCHAR(20) PRIMA...	0 row(s) affected
✓ 4	11:21:59	CREATE TABLE IF NOT EXISTS companies ( company_id VARCHAR(20...	0 row(s) affected

CREATE TABLE IF NOT EXISTS **users** (  
 id INT PRIMARY KEY,  
 name VARCHAR(50) NOT NULL,  
 surname VARCHAR(50) NOT NULL,  
 phone VARCHAR(20) UNIQUE,  
 email VARCHAR(50) UNIQUE NOT NULL,  
 birth\_date VARCHAR(20),  
 country VARCHAR(20),  
 city VARCHAR(20),  
 postal\_code VARCHAR(20),  
 adress VARCHAR(100));

```

29
30 CREATE TABLE IF NOT EXISTS users (
31     id INT PRIMARY KEY,
32     name VARCHAR(50) NOT NULL,
33     surname VARCHAR(50) NOT NULL,
34     phone VARCHAR(20) UNIQUE,
35     email VARCHAR(50) UNIQUE NOT NULL,
36     birth_date VARCHAR(10),
37     country VARCHAR(20),
38     city VARCHAR(20),
39     postal_code VARCHAR(20),
40     address VARCHAR(100));
41

```

Output

#	Time	Action	Message
21	12:21:50	DROP TABLE users	0 row(s) affected
22	12:21:58	CREATE TABLE IF NOT EXISTS users (id...	0 row(s) affected

CREATE TABLE IF NOT EXISTS **transactions** (  
 id VARCHAR(100) PRIMARY KEY,  
 card\_id VARCHAR(20) NOT NULL,  
 business\_id VARCHAR(20) NOT NULL,  
 timestamp TIMESTAMP,  
 amount DECIMAL(10,2) NOT NULL,  
 declined boolean,  
 product\_ids VARCHAR(50) NOT NULL,  
 user\_id INT NOT NULL,  
 lat FLOAT,  
 longitude FLOAT);

```

42 CREATE TABLE IF NOT EXISTS transactions (
43     id VARCHAR(100) PRIMARY KEY,
44     card_id VARCHAR(20) NOT NULL,
45     business_id VARCHAR(20) NOT NULL,
46     timestamp TIMESTAMP,
47     amount DECIMAL(10,2) NOT NULL,
48     declined boolean,
49     product_ids VARCHAR(50) NOT NULL,
50     user_id INT NOT NULL,
51     lat FLOAT,
52     longitude FLOAT);
53

```

Output

#	Time	Action	Message
5	11:23:28	CREATE TABLE IF NOT EXISTS users (id INT PRIMARY KEY, name ...	0 row(s) affected
6	11:23:43	CREATE TABLE IF NOT EXISTS transactions (id VARCHAR(100) PRIM...	0 row(s) affected

### Inserción de datos:

Para poder insertar un archivo local para insertar los datos hay que verificar si la opción está activada. La respuesta inicial es negativa, aparece como OFF, por lo que la activamos. Comprobamos después de activarla y aparece como ON.

```

SHOW VARIABLES LIKE 'local_infile';
SET GLOBAL local_infile = 1;

```

```

55 • SHOW VARIABLES LIKE 'local_infile';
56 • SET GLOBAL local_infile = 1;

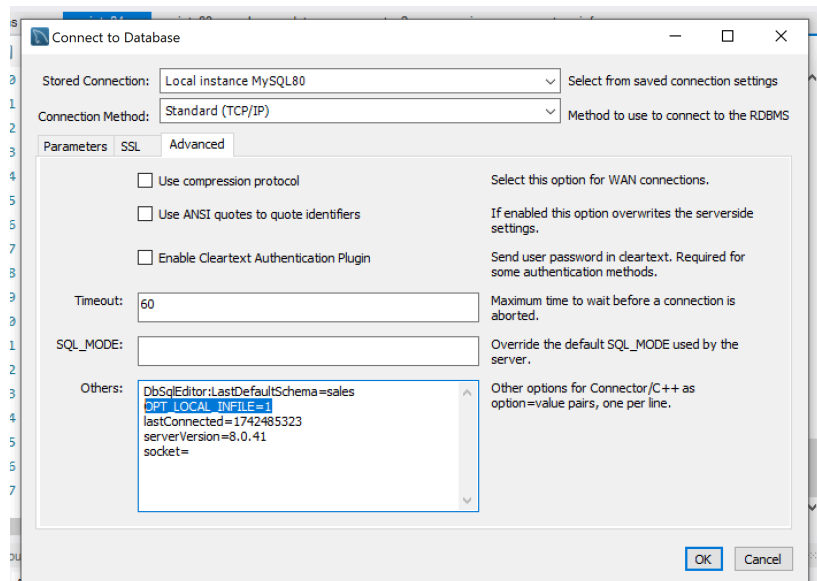
```

Variable_name	Value
local_infile	ON

#	Time	Action	Message
7	11:25:00	SHOW VARIABLES LIKE 'local_infile'	1 row(s) returned
8	11:25:33	SET GLOBAL local_infile = 1	0 row(s) affected
9	11:25:43	SHOW VARIABLES LIKE 'local_infile'	1 row(s) returned

También hemos tenido que ir desde la barra de herramientas de Mysql a 'Database' a 'Manage connections' y en la pestaña 'Advanced' añadir en 'Others' el código: OPT\_LOCAL\_INFILE=1.



**Insertamos los datos:**

```

LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

SELECT * FROM credit_cards;

```

```

58 -- inserción de datos
59 • LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/credit_cards.csv'
60 INTO TABLE credit_cards
61 FIELDS TERMINATED BY ','
62 OPTIONALLY ENCLOSED BY '"'
63 LINES TERMINATED BY '\n'
64 IGNORE 1 ROWS;
65
66 • SELECT * FROM credit_cards;

```

id	user_id	iban	pan	pin	cvv	track1
CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%B8383712448554646^Wc
CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%B4621311609958661^Uft
CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%B2183285104307501^Ccd
CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%B7281111956795320^Xor

credit\_cards 1 x

Output

Action Output

#	Time	Action	Message
✓ 14	11:59:28	LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/credit_cards.csv' INTO TABLE credit_cards	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0
✓ 15	12:13:30	SELECT * FROM credit_cards LIMIT 0, 1000	275 row(s) returned

LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint\_04/companies.csv'  
 INTO TABLE **companies**  
 FIELDS TERMINATED BY ','  
 OPTIONALLY ENCLOSED BY '"'  
 LINES TERMINATED BY '\n'  
 IGNORE 1 ROWS;

SELECT \* FROM companies;

```

68 • LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/companies.csv'
69 INTO TABLE companies
70 FIELDS TERMINATED BY ','
71 OPTIONALLY ENCLOSED BY '"'
72 LINES TERMINATED BY '\n'
73 IGNORE 1 ROWS;
74
75 • SELECT * FROM companies;

```

company_id	company_name	phone	email	country	website
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://inc
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://wi
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pir
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.co.uk	Germany	https://cn

companies 2 x

Output

Action Output

#	Time	Action	Message
✓ 17	12:18:31	LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/companies.csv' INTO TABLE companies	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
✓ 18	12:18:40	SELECT * FROM companies LIMIT 0, 1000	100 row(s) returned

Para la inserción de los datos user, daba error al insertar datos, al principio pensábamos que era porque había campos que contenían comas y llevaba a confusión, pero al mirar el excel hemos visto que la separación era correcta. Por lo tanto, hemos pensado que el problema podría radicar en el salto de línea y al cambiar \n a \r\n ha funcionado. Los archivos de credit\_cards, companies y transactions han sido creados en formato Unix/Mac OS X y el de users en formato Windows. A continuación específico las diferencias entre los diferentes tipos de saltos de línea.

\r = CR (Retorno de carro) → Se usa como carácter de nueva línea en Mac OS antes de X  
 \n = LF (Salto de línea) → Se usa como carácter de nueva línea en Unix/Mac OS X

$\text{\r\n} = \text{CR} + \text{LF} \rightarrow$  Se usa como carácter de nueva línea en Windows

```
LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/users_ca.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

```
LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/users_uk.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

```
LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/users_usa.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

```
SELECT * FROM users;
```

The screenshot shows a SQL IDE interface. The top pane displays the following SQL commands:

```
91 • LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/users_
92 INTO TABLE users
93 FIELDS TERMINATED BY ','
94 ENCLOSED BY '"'
95 LINES TERMINATED BY '\r\n'
96 IGNORE 1 ROWS;
97
98 • SELECT * FROM users;
```

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

	id	name	surname	phone	email	birth_date	country
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United St
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United St
	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United St
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United St
	5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United St
	6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United St

Below the result grid, there is an 'Output' section showing the execution log:

#	Time	Action	Message
✓ 87	13:34:30	LOAD DATA LOCAL...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0
✓ 88	13:35:01	LOAD DATA LOCAL...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0
✓ 89	13:35:31	LOAD DATA LOCAL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0
✓ 90	13:37:17	SELECT * FROM us...	275 row(s) returned

```
LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

```
SELECT * FROM transactions;
```

```

100 • LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/transactions.csv'
101 INTO TABLE transactions
102 FIELDS TERMINATED BY ';'
103 ENCLOSED BY ''
104 LINES TERMINATED BY '\n'
105 IGNORE 1 ROWS;
106
107 • SELECT * FROM transactions;

```

id	card_id	business_id	timestamp	amount	declined	product
02C6201E-D90A-1859-84EE-88D2986D3802	CdJ-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 15
0466A42E-47CF-8D24-FD01-C0B689713128	CdJ-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 4
063FBA79-99EC-66FB-29F7-25726D1764A5	CdJ-2987	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 3
0668296C-CDB9-A883-76BC-2E4C4F8C8AE	CdJ-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 7

transactions 3 x Apply

Output

Action Output

#	Time	Action	Message
53	12:47:49	LOAD D...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0
54	12:47:53	SELECT...	587 row(s) returned

## Modificación en las tablas:

### CREDIT\_CARDS

Eliminamos el campo credit\_card.user\_id:

ALTER TABLE credit\_card DROP COLUMN user\_id;

```

109 • ALTER TABLE credit_cards DROP COLUMN user_id;
110 • SHOW COLUMNS FROM credit_cards;

```

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	NO	UNI	NULL	
pan	varchar(50)	NO	UNI	NULL	
pin	varchar(50)	NO		NULL	
cvv	int	NO		NULL	
track1	varchar(50)	YES		NULL	
track2	varchar(50)	YES		NULL	
expiring_date	varchar(10)	NO		NULL	

Result 50 x

Output

Action Output

#	Time	Action	Message
294	16:32:48	ALTER TABLE credit_cards DROP COLU...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
295	16:33:31	SHOW COLUMNS FROM credit_cards	8 row(s) returned

### Modificamos el tipo de dato de la columna expiring\_date:

ALTER TABLE credit\_cards ADD COLUMN expiring\_date\_temp DATE;

UPDATE credit\_cards SET expiring\_date\_temp = STR\_TO\_DATE(expiring\_date, '%m/%d/%y');

ALTER TABLE credit\_cards DROP expiring\_date;

ALTER TABLE credit\_cards CHANGE expiring\_date\_temp expiring\_date DATE NOT NULL;

SHOW COLUMNS FROM credit\_cards;

```

111 -- Modificamos el tipo de dato de la columna expiring_date:
112 • ALTER TABLE credit_cards ADD COLUMN expiring_date_temp DATE;
113 • UPDATE credit_cards SET expiring_date_temp = STR_TO_DATE(expiring_date, '%m/%d/%y');
114 • ALTER TABLE credit_cards DROP expiring_date;
115 • ALTER TABLE credit_cards CHANGE expiring_date_temp expiring_date DATE NOT NULL;
116 • SHOW COLUMNS FROM credit_cards;
117

```

Result 52

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	NO	UNI	NULL	
pan	varchar(50)	NO	UNI	NULL	
pin	varchar(50)	NO		NULL	
cvv	int	NO		NULL	
track1	varchar(50)	YES		NULL	
track2	varchar(50)	YES		NULL	
expiring_date	date	NO		NULL	

Output

Action Output

#	Time	Action	Message
312	16:44:35	ALTER TABLE credit_cards ADD COLUMN...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
313	16:44:38	UPDATE credit_cards SET expiring_date_...	275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0
314	16:44:41	ALTER TABLE credit_cards DROP expirin...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
315	16:44:45	ALTER TABLE credit_cards CHANGE expi...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
316	16:44:47	SHOW COLUMNS FROM credit_cards	8 row(s) returned

## USERS

**En user modificamos el tipo de dato de la columna birth\_date:**

ALTER TABLE user ADD COLUMN birth\_date\_temp DATE;

UPDATE user SET birth\_date\_temp = STR\_TO\_DATE(birth\_date, '%b %d, %Y');

ALTER TABLE user DROP birth\_date;

ALTER TABLE user CHANGE birth\_date\_temp birth\_date DATE NOT NULL;

```

112 -- En user modificamos el tipo de dato de la columna birth_date:
113 • ALTER TABLE users ADD COLUMN birth_date_temp DATE;
114 • UPDATE users SET birth_date_temp = STR_TO_DATE(birth_date, '%b %d, %Y');
115 • ALTER TABLE users DROP birth_date;
116 • ALTER TABLE users CHANGE birth_date_temp birth_date DATE NOT NULL;
117 • SHOW COLUMNS FROM users;

```

Result 51

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
surname	varchar(50)	YES		NULL	
phone	varchar(20)	YES		NULL	
email	varchar(255)	YES		NULL	
country	varchar(100)	YES		NULL	
city	varchar(100)	YES		NULL	
postal_code	varchar(50)	YES		NULL	
adress	varchar(255)	YES		NULL	
birth_date	date	NO		NULL	

Output

Action Output

#	Time	Action	Message
303	16:37:00	ALTER TABLE users ADD COLUMN birth...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
304	16:37:03	UPDATE users SET birth_date_temp = ST...	275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0
305	16:37:07	ALTER TABLE users DROP birth_date	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
306	16:37:10	ALTER TABLE users CHANGE birth_date...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
307	16:37:13	SHOW COLUMNS FROM users	10 row(s) returned



**Mostrar que se ha creado correctamente:**

**La base de datos:**

SHOW TABLES FROM sales;

116 • SHOW TABLES FROM sales;

117

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

Tables\_in\_sales

- companies
- credit\_cards
- transactions
- users

Result 8 x

Output

Action Output

#	Time	Action	Message
✓ 97	13:41:27	ALTER TABLE credit_...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
✓ 98	13:45:23	SHOW TABLES FRO...	4 row(s) returned

**Las tablas:**

SHOW COLUMNS FROM credit\_cards;

128 • SHOW COLUMNS FROM credit\_cards;

129 • SHOW COLUMNS FROM companies;

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

Field	Type	Null	Key	Default	Extra
id	varchar(20)	NO	PRI	NULL	
iban	varchar(50)	NO	UNI	NULL	
pan	varchar(50)	NO	UNI	NULL	
pin	varchar(50)	NO		NULL	
cvv	int	NO		NULL	
track1	varchar(50)	YES		NULL	
track2	varchar(50)	YES		NULL	
expiring_date	date	NO		NULL	

Result 52 x

SHOW COLUMNS FROM companies;

129 • SHOW COLUMNS FROM companies;

130 • SHOW COLUMNS FROM users;

Result Grid | Filter Rows: | Export: | Wrap C

Field	Type	Null	Key	Default	Extra
company_id	varchar(20)	NO	PRI	NULL	
company_name	varchar(50)	NO		NULL	
phone	varchar(20)	YES	UNI	NULL	
email	varchar(50)	YES		NULL	
country	varchar(20)	YES		NULL	
website	varchar(100)	YES		NULL	

SHOW COLUMNS FROM users;

130 • SHOW COLUMNS FROM users;

Result Grid | Filter Rows: | Export: |

Field	Type	Null	Key	Default
id	int	NO	PRI	NULL
name	varchar(50)	YES		NULL
surname	varchar(50)	YES		NULL
phone	varchar(20)	YES		NULL
email	varchar(255)	YES		NULL
country	varchar(100)	YES		NULL
city	varchar(100)	YES		NULL
postal_code	varchar(50)	YES		NULL
adress	varchar(255)	YES		NULL
birth_date	date	NO		NULL

SHOW COLUMNS FROM transactions;

121 • SHOW COLUMNS FROM transactions;

122

<

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

Field	Type	Null	Key	Default	Extra
id	varchar(100)	NO	PRI	NULL	
card_id	varchar(20)	NO	MUL	NULL	
business_id	varchar(20)	NO	MUL	NULL	
timestamp	timestamp	YES		NULL	
amount	decimal(10,2)	NO		NULL	
declined	tinyint(1)	YES		NULL	
product_ids	varchar(50)	NO		NULL	
user_id	int	NO	MUL	NULL	
lat	float	YES		NULL	
longitude	float	YES		NULL	

Result 16 x

Output

Action Output

#	Time	Action	Message
✓ 106	13:53:01	SHOW COLUMNS FROM users	10 row(s) returned
✓ 107	13:53:52	SHOW COLUMNS FROM transactions	10 row(s) returned

### Creamos las relaciones de Foreign keys entre las tablas

ALTER TABLE transactions

ADD CONSTRAINT fk\_transactions\_credit\_cards FOREIGN KEY (card\_id) REFERENCES credit\_cards (id),

ADD CONSTRAINT fk\_transactions\_companies FOREIGN KEY (business\_id) REFERENCES companies (company\_id),

ADD CONSTRAINT fk\_transactions\_users FOREIGN KEY (user\_id) REFERENCES users (id);

### Comprobamos que se han creado correctamente

SHOW INDEXES FROM transactions;

134 -- Creamos las relaciones de Foreign keys entre las tablas

135 • ALTER TABLE transactions

136 ADD CONSTRAINT fk\_transactions\_credit\_cards FOREIGN KEY (card\_id) REFERENCES credit\_cards (id),

137 ADD CONSTRAINT fk\_transactions\_companies FOREIGN KEY (business\_id) REFERENCES companies (company\_id),

138 ADD CONSTRAINT fk\_transactions\_users FOREIGN KEY (user\_id) REFERENCES users (id);

139

140 -- Comprobamos que se han creado correctamente

141 • SHOW INDEXES FROM transactions;

142

<

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

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed
transactions	0	PRIMARY	1	id	A	587	NULL	NULL
transactions	1	fk_transactions_credit_cards	1	card_id	A	275	NULL	NULL
transactions	1	fk_transactions_companies	1	business_id	A	100	NULL	NULL
transactions	1	fk_transactions_users	1	user_id	A	216	NULL	NULL

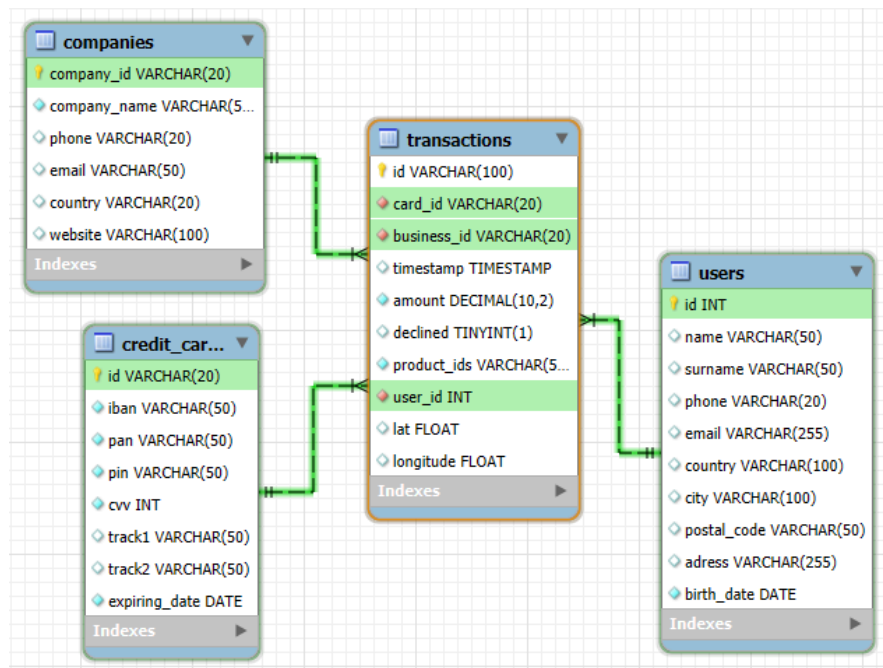
Result 56 x

Output

Action Output

#	Time	Action	Message
✓ 324	16:54:29	ALTER TABLE transactions ADD CONST...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
✓ 325	16:54:44	SHOW INDEXES FROM transactions	4 row(s) returned

## Esquema final de la base de datos 'sales':



## - Exercici 1

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

Seleccionamos el id de usuario y el nombre y apellido que con el método CONCAT unimos para que aparezcan juntos en una sola columna, filtramos a través de una subconsulta donde seleccionamos y agrupamos por el id de usuario y filtramos a los usuarios con más de 30 transacciones con un COUNT y HAVING.

```
SELECT users.id AS id_usuario, CONCAT(users.name, " ", users.surname) AS nombre_usuario
FROM users
WHERE id IN (SELECT user_id FROM transactions
              GROUP BY user_id
              HAVING COUNT(id) > 30);
```

```

124 -- Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.
125 • SELECT users.id AS id_usuario, CONCAT(users.name, " ", users.surname) AS nombre_usuario
126 FROM users
127 WHERE id IN (SELECT user_id FROM transactions
128              GROUP BY user_id
129              HAVING COUNT(id) > 30);

```

id_usuario	nombre_usuario
92	Lynn Riddle
267	Ocean Nelson
272	Hedwig Gilbert
275	Kenyon Hartman

Result 8 x

Output

#	Time	Action	Message
15	17:27:18	SELECT users.id AS id_usuario, CONCAT(users.name, " ", users.surname) AS nombre_...	Error Code: 1054. Unknown column 'num_transacciones' in 'field list'
16	17:27:59	SELECT users.id AS id_usuario, CONCAT(users.name, " ", users.surname) AS nombre_...	4 row(s) returned

## - Exercici 2

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

Seleccionamos el iban y hacemos la media del amount redondeada. Unimos las tablas transactions, credit\_cards y companies por id de tarjeta e id de empresa con un LEFT JOIN. Filtramos con WHERE por el nombre de empresa deseado y agrupamos por iban.

-- Opción 1: teniendo en cuenta todas las transacciones

SELECT iban, ROUND(AVG(amount),2) AS media\_ventas

FROM transactions

LEFT JOIN credit\_cards ON transactions.card\_id = credit\_cards.id

LEFT JOIN companies ON transactions.business\_id = companies.company\_id

WHERE company\_name = "Donec Ltd"

GROUP BY iban;

```

134 -- Opción 1: teniendo en cuenta todas las transacciones
135 • SELECT iban, ROUND(AVG(amount),2) AS media_ventas
136 FROM transactions
137 LEFT JOIN credit_cards ON transactions.card_id = credit_cards.id
138 LEFT JOIN companies ON transactions.business_id = companies.company_id
139 WHERE company_name = "Donec Ltd"
140 GROUP BY iban;

```

iban	media_ventas
PT87806228135092429456346	203.72

10 17:13:34 SELECT iban, ROUND(AVG(amount),2) AS media\_ventas FROM transactions LEFT JOIN credit\_cards ON trans... 1 row(s) returned

Opción 2: hacemos lo mismo, pero descartando las transacciones con pagos rechazados

SELECT iban, ROUND(AVG(amount),2) AS media\_ventas

FROM transactions

JOIN credit\_cards ON transactions.card\_id = credit\_cards.id

JOIN companies ON transactions.business\_id = companies.company\_id

WHERE company\_name = "Donec Ltd" AND declined = 0

GROUP BY iban;

Obtenemos un resultado distinto:

```
142 -- Opción 2: hacemos lo mismo, pero descartando las transacciones con pagos rechazados
143 • SELECT iban, ROUND(AVG(amount),2) AS media_ventas
144 FROM transactions
145 LEFT JOIN credit_cards ON transactions.card_id = credit_cards.id
146 LEFT JOIN companies ON transactions.business_id = companies.company_id
147 WHERE company_name = "Donec Ltd" AND declined = 0
148 GROUP BY iban;
```

iban	media_ventas
PT87806228135092429456346	42.82

12 17:18:36 SELECT iban, ROUND(AVG(amount),2) AS media\_ventas FROM transactions LEFT JOIN credit\_cards ON trans... 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 i genera la següent consulta:*

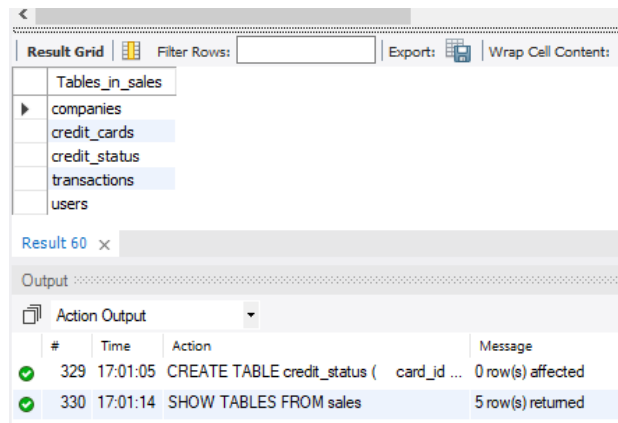
### Creemos la nueva tabla que es actualizable:

Creemos la nueva tabla llamada `credit_status` que muestre el estado como activa o bloqueada de las tarjetas basado a si en las últimas 3 transacciones han sido declinadas o no.

Utilizamos `card_id` tipo `VARCHAR` como Primary Key (mismo que el id de la tabla `credit_cards` con la que la relacionamos), el `iban` tipo `VARCHAR` como en la tabla `credit_cards` y creamos una columna nueva llamada `status` que podría ser tipo Boolean, pero hemos pensado que mejor tipo `ENUM` ya que permite mostrar elementos determinados, en este caso dos grupos: 'activa' para las tarjetas que han pasado el filtro como que no han sido declinadas en las últimas 3 transacciones y 'bloqueada' para aquellas tarjetas que han pasado el filtro como que sí tienen sus últimas 3 transacciones declinadas. Relacionamos la tabla 1 a N con la tabla `credit_cards`. Comprobamos que se ha realizado correctamente

```
CREATE TABLE credit_status (
  card_id VARCHAR(20) PRIMARY KEY,
  status ENUM('activa', 'bloqueada') NOT NULL,
  FOREIGN KEY (card_id) REFERENCES credit_cards(id));
```

```
174 -- creamos la tabla
175 • CREATE TABLE credit_status (
176   card_id VARCHAR(20) PRIMARY KEY,
177   status ENUM('activa', 'bloqueada') NOT NULL);
178
179 -- Comprobamos que se ha creado correctamente
180 • SHOW TABLES FROM sales;
```



### Hacemos la consulta para identificar las transacciones declinadas e insertar en la tabla:

Insertamos en la tabla credit\_status los datos en las columnas pertinentes: card\_id y status que sacamos de la siguiente consulta y utilizando para insertar entre bloqueada y activa mediante el método CASE.

Para la consulta utilizamos DENSE\_RANK() para ordenar las transacciones según el timestamp descendientemente y particionando por el card\_id. A esto lo llamamos rango. A esta subconsulta le filtramos que seleccione las primeras 3. Al hacer el CASE filtramos los declinados donde cogemos la suma de los declinados del rango limitado a 3. Por lo que aquellos que tengan menos de 3 (3 transacciones últimas declinadas) son consideradas 'activa' en status. Aquellas card\_id que tengan la suma de declinados = 3 serán entonces marcadas como 'bloqueada' en status.

```
INSERT INTO credit_status (card_id, status)
SELECT card_id,
CASE WHEN sum(declined) < 3
      THEN 'activa'
      ELSE 'bloqueada'
END AS status
FROM (SELECT transactions.card_id, transactions.timestamp, declined,
      DENSE_RANK() OVER (PARTITION BY transactions.card_id
                        ORDER BY transactions.timestamp DESC) AS rango
      FROM transactions) AS consulta_rango
WHERE rango <= 3
GROUP BY card_id
ORDER BY card_id;
```

```

174 -- creamos la tabla
175 • CREATE TABLE credit_status (
176     card_id VARCHAR(20) PRIMARY KEY,
177     status ENUM('activa', 'bloqueada') NOT NULL);
178
179 -- Comprobamos que se ha creado correctamente
180 • SHOW TABLES FROM sales;
181
182 -- Insertar en la tabla los datos filtrados
183 • INSERT INTO credit_status (card_id, status)
184     SELECT card_id,
185     CASE WHEN sum(declined) < 3
186     THEN 'activa'
187     ELSE 'bloqueada'
188     END AS status
189     FROM (SELECT transactions.card_id, transactions.timestamp, declined,
190     DENSE_RANK() OVER (PARTITION BY transactions.card_id
191     ORDER BY transactions.timestamp DESC) AS rango
192     FROM transactions) AS consulta_rango
193     WHERE rango <= 3
194     GROUP BY card_id
195     ORDER BY card_id;
196
197 <

```

Output

#	Time	Action	Message
143	17:24:42	INSERT INTO credit_status (card_id, iban, status) SELECT card_id, CASE WHEN sum(declined) < 3 THEN 'a'...	Error Code: 1054, Unknown column 'iban' in field list
144	17:25:09	INSERT INTO credit_status (card_id, status) SELECT card_id, CASE WHEN sum(declined) < 3 THEN 'activa' ...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

## Comprobación:

Efectivamente, las 275 tarjetas que hay están activas.

```

191     ORDER BY transactions.timestamp DESC) AS rango
192     FROM transactions) AS consulta_rango
193     WHERE rango <= 3
194     GROUP BY card_id
195     ORDER BY card_id;
196
197 • SELECT * FROM credit_status;

```

Result Grid

card_id	status
CcU-2938	activa
CcU-2945	activa
CcU-2952	activa
CcU-2959	activa
CcU-2966	activa
CcU-2973	activa
CcU-2980	activa
CcU-2987	activa
CcU-2994	activa
CcU-2995	activa
CcU-2996	activa
CcU-2997	activa
CcU-2998	activa
CcU-2999	activa
CcU-3000	activa
CcU-3001	activa
CcU-3002	activa
CcU-3003	activa
CcU-3004	activa
CcU-3005	activa
CcU-3006	activa
CcU-3007	activa
CcU-3008	activa
CcU-3009	activa
CcU-3010	activa
CcU-3011	activa
CcU-3012	activa
CcU-3013	activa
CcU-3014	activa
CcU-3015	activa
CcU-3016	activa
CcU-3017	activa
CcU-3018	activa
CcU-3019	activa
CcU-3020	activa
CcU-3021	activa
CcU-3022	activa
CcU-3023	activa
CcU-3024	activa
CcU-3025	activa
CcU-3026	activa
CcU-3027	activa
CcU-3028	activa
CcU-3029	activa
CcU-3030	activa
CcU-3031	activa
CcU-3032	activa
CcU-3033	activa
CcU-3034	activa
CcU-3035	activa
CcU-3036	activa
CcU-3037	activa
CcU-3038	activa
CcU-3039	activa
CcU-3040	activa
CcU-3041	activa
CcU-3042	activa
CcU-3043	activa
CcU-3044	activa
CcU-3045	activa
CcU-3046	activa
CcU-3047	activa
CcU-3048	activa
CcU-3049	activa
CcU-3050	activa
CcU-3051	activa
CcU-3052	activa
CcU-3053	activa
CcU-3054	activa
CcU-3055	activa
CcU-3056	activa
CcU-3057	activa
CcU-3058	activa
CcU-3059	activa
CcU-3060	activa
CcU-3061	activa
CcU-3062	activa
CcU-3063	activa
CcU-3064	activa
CcU-3065	activa
CcU-3066	activa
CcU-3067	activa
CcU-3068	activa
CcU-3069	activa
CcU-3070	activa
CcU-3071	activa
CcU-3072	activa
CcU-3073	activa
CcU-3074	activa
CcU-3075	activa
CcU-3076	activa
CcU-3077	activa
CcU-3078	activa
CcU-3079	activa
CcU-3080	activa
CcU-3081	activa
CcU-3082	activa
CcU-3083	activa
CcU-3084	activa
CcU-3085	activa
CcU-3086	activa
CcU-3087	activa
CcU-3088	activa
CcU-3089	activa
CcU-3090	activa
CcU-3091	activa
CcU-3092	activa
CcU-3093	activa
CcU-3094	activa
CcU-3095	activa
CcU-3096	activa
CcU-3097	activa
CcU-3098	activa
CcU-3099	activa
CcU-3100	activa

Output

#	Time	Action	Message
144	17:25:09	INSERT INTO credit_status (card_id, status) SELECT card_id, CASE WHEN sum(declined) < 3 THEN 'activa' ...	275 row(s) affected Records
145	17:36:23	SELECT * FROM credit_status LIMIT 0, 1000	275 row(s) returned

## Añadimos el Foreign Key

ALTER TABLE credit\_status

ADD CONSTRAINT fk\_credit\_status\_credit\_cards FOREIGN KEY (card\_id) REFERENCES credit\_cards (id);

SHOW INDEXES FROM credit\_status;

```

200 -- Añadimos el Foreign Key
201 • ALTER TABLE credit_status
202 ADD CONSTRAINT fk_credit_status_credit_cards FOREIGN KEY (card_id) REFERENCES credit_cards (id);
203
204 • SHOW INDEXES FROM credit_status;
205

```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visit
credit_status	0	PRIMARY	1	card_id	A	0			NULL	BTREE			YES

#	Time	Action	Message
✓ 146	17:44:12	ALTER TABLE credit_status ADD CONSTRAINT fk_credit_status_credit_cards FOREIGN KEY (card_id) REF...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
✓ 147	17:44:30	SHOW INDEXES FROM credit_status	1 row(s) returned

## Exercici 1

*Quantes targetes estan actives?*

```

SELECT COUNT(*) AS targetas_activas
FROM credit_card_status
WHERE status = 'activa';

```

```

206 -- Ejercicio 1
207 -- Quantes targetes estan actives?
208 • SELECT COUNT(*) AS targetas_activas
209 FROM credit_status
210 WHERE status = 'activa';

```

targetas_activas
275

#	Time	Action	Message
✓ 147	17:44:30	SHOW INDEXES FROM credit_status	1 row(s) returned
✓ 148	17:45:48	SELECT COUNT(*) AS targetas_activas FROM credit_status WHERE status = 'activa' LIMIT 0, 1000	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. Genera la següent consulta:*

**Primero crearemos la tabla products:**

```

CREATE TABLE IF NOT EXISTS products (
  id INT PRIMARY KEY,
  product_name VARCHAR(50) NOT NULL,
  price VARCHAR(10) NOT NULL,

```



color VARCHAR(20),  
weight FLOAT,  
warehouse\_id VARCHAR(10));

```

185 -- Primero crearemos la tabla products:
186 • CREATE TABLE IF NOT EXISTS products (
187     id INT PRIMARY KEY,
188     product_name VARCHAR(50) NOT NULL,
189     price VARCHAR(10) NOT NULL,
190     color VARCHAR(20),
191     weight FLOAT,
192     warehouse_id VARCHAR(10));
193

```

Output

#	Time	Action	Message
95	12:23:15	SELECT transactions.timestamp, t...	587 row(s) returned
96	12:30:26	CREATE TABLE IF NOT EXISTS ...	0 row(s) affected

### Insertamos los datos en la tabla products:

LOAD DATA INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint\_04/products.csv'  
INTO TABLE products  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''''  
LINES TERMINATED BY '\r\n'  
IGNORE 1 ROWS;

```

202 -- Insertamos los datos en la tabla products:
203 • LOAD DATA LOCAL INFILE 'C:/Users/Nicola Korff/Desktop/SQL/da/sprint_04/products.cs
204 INTO TABLE products
205 FIELDS TERMINATED BY ','
206 ENCLOSED BY ''''
207 LINES TERMINATED BY '\r\n'
208 IGNORE 1 ROWS;
209
210 • SELECT * FROM products;

```

Result Grid

	id	product_name	price	color	weight	warehouse_id
▶	1	Direwolf Stannis	\$161.11	#7c7c7c	1	WH-4
	2	Tarly Stark	\$9.24	#919191	2	WH-3
	3	duel tourney Lannister	\$171.13	#d8d8d8	1.5	WH-2
	4	warden south duel	\$71.89	#111111	3	WH-1
	5	skywalker ewok	\$171.22	#bdbdbd	3.2	WH-0
	6	dooku solo	\$136.60	#c4c4c4	0.8	WH--1
	7	north of Casterly	\$63.33	#b7b7b7	0.6	WH--2
	8	Winterfell	\$37.37	#383838	1.4	WH--3

products 20 x Apply

Output

#	Time	Action	Message
103	12:39:29	LOAD DATA LOCAL INFILE 'C:/Use...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
104	12:40:59	SELECT * FROM products LIMIT 0, ...	100 row(s) returned

Comprobamos que se ha creado correctamente:

SHOW TABLES FROM sales;

SHOW COLUMNS FROM products;

```

232 -- Comprobamos que se ha cre
233 • SHOW TABLES FROM sales;
234

```

Result Grid

Filter Rows:

Tables_in_sales
companies
credit_cards
credit_status
products
transactions
users

```

234 • SHOW COLUMNS FROM products;
235

```

Result Grid

Filter Rows:

Export:

Wrap

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	
product_name	varchar(50)	NO		NULL	
price	varchar(10)	NO		NULL	
color	varchar(20)	YES		NULL	
weight	float	YES		NULL	
warehouse_id	varchar(10)	YES		NULL	

En la columna price separamos el símbolo de dolar a una nueva columna denominada **currency**:

ALTER TABLE products ADD COLUMN currency VARCHAR(1);

UPDATE products

SET currency = LEFT(price, 1),

price = CAST(SUBSTRING(price, 2) AS DECIMAL(10,2));

```

239 -- En la columna price separamos el simbolo de dolar a una nueva columna denominada currency:
240 • ALTER TABLE products ADD COLUMN currency VARCHAR(1);
241 • UPDATE products
242     SET currency = LEFT(price, 1),
243     price = CAST(SUBSTRING(price, 2) AS DECIMAL(10,2));
244
245 -- Comprobamos como ha quedado:
246 • SELECT * FROM products;

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Contents:

id	product_name	price	color	weight	warehouse_id	currency
1	Direwolf Stannis	161.11	#7c7c7c	1	WH-4	\$
2	Tarly Stark	9.24	#919191	2	WH-3	\$
3	duel tourney Lannister	171.13	#d8d8d8	1.5	WH-2	\$
4	warden south duel	71.89	#111111	3	WH-1	\$
5	skywalker ewok	171.22	#dbdbdb	3.2	WH-0	\$

products 34 x

Output

Action Output

#	Time	Action	Message
153	17:55:09	UPDATE products SET currency = LEFT(price, 1), price = CAST(SUBSTRING(price, 2) AS DECIMAL(10,2))	100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0
154	17:55:49	SELECT * FROM products LIMIT 0, 1000	100 row(s) returned

Para integrar los datos de products.csv en la base de datos y relacionarlos con transactions.product\_ids, necesitamos una tabla intermedia para gestionar la relación entre transacciones y productos, ya que product\_ids puede contener múltiples valores por transacción en un mismo campo.

Crearemos la tabla orders desde transacciones importando los el id de la transacción y los productos de la columna products\_id:

CREATE TABLE orders SELECT id, products\_id FROM transaction;

**Comprobamos:**

SELECT \* FROM orders;

```

248 -- Creamos la tabla intermedia a la que denominamos orders a partir de las columnas id y product_ids de transactions:
249 • CREATE TABLE orders SELECT id, product_ids FROM transactions;
250
251 • SELECT * FROM orders;

```

id	product_ids
02C6201E-D90A-1859-B4EE-88D2986D3802	71, 1, 19
0466A42E-47CF-8D24-FD01-C0B689713128	47, 97, 43
063FBA79-99EC-66FB-29F7-25726D1764A5	47, 67, 31, 5
0668296C-CDB9-A883-76BC-2E4C4F8C8AE	89, 83, 79
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43, 31

orders 35 x

Output

#	Time	Action	Message
155	18:00:21	CREATE TABLE orders SELECT id, product_ids FROM transactions	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
156	18:00:25	SELECT * FROM orders LIMIT 0, 1000	587 row(s) returned

Creamos una tabla temporal a la que denominamos temp\_orders donde podemos insertar los valores de los products\_id separados por comas

```

CREATE TABLE temp_orders (
    id VARCHAR(100),
    product_id1 INT,
    product_id2 INT,
    product_id3 INT,
    product_id4 INT );

```

```

254 -- creamos la tabla temporal:
255 • CREATE TABLE temp_orders (
256     id VARCHAR (100),
257     product_id1 INT,
258     product_id2 INT,
259     product_id3 INT,
260     product_id4 INT);
261

```

Output

#	Time	Action	Message
156	18:00:25	SELECT * FROM orders LIMIT 0, 1000	587 row(s) returned
157	18:02:46	CREATE TABLE temp_orders (id VARCHAR(100), product_id1 INT, product_id2 INT, product_id3 INT, pro...	0 row(s) affected

Insertamos los datos en la tabla temps\_orders, extrayendo y separando los valores de la columna product\_ids de la tabla orders que acabamos de crear, que contiene los ids de los productos separados por comas como en la tabla transactions. Esto lo haremos mediante la función SUBSTRING\_INDEX doble. El último argumento de la función que va de 1 a 4 devuelve los elementos de la fila. 1 devuelve el primer elemento; 2 obtiene los primeros 2 elementos y con -1 tomamos el último de los que obtuvimos; lo mismo con 3 y 4. Es decir, cada fila de orders se transformará en una fila en temp\_orders, donde los valores de product\_ids se dividirán en hasta 4 columnas. En caso de no haber valor se convierte en NULL. Con CAST... AS UNSIGNED transformamos los valores extraídos de tipo VARCHAR y los convierte en números enteros (UNSIGNED) para que se almacenen correctamente en la tabla.

```

INSERT INTO temp_orders (id, product_id1, product_id2, product_id3, product_id4)
SELECT id,
    CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 1), ',', -1) AS UNSIGNED) AS
product_id1,
    CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1) AS UNSIGNED) AS
product_id2,

```

```

CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 3), ',', -1) AS UNSIGNED) AS
product_id3,
CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 4), ',', -1) AS UNSIGNED) AS
product_id4
FROM orders;

```

```

262 -- insertamos los productos_id de la tabla
263 • INSERT INTO temp_orders (id, product_id1, product_id2, product_id3, product_id4)
264 SELECT id,
265     CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 1), ',', -1) AS UNSIGNED) AS product_id1,
266     CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1) AS UNSIGNED) AS product_id2,
267     CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 3), ',', -1) AS UNSIGNED) AS product_id3,
268     CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 4), ',', -1) AS UNSIGNED) AS product_id4
269 FROM orders;
270
271 • SELECT * FROM temp_orders;

```

id	product_id1	product_id2	product_id3	product_id4
02C6201E-D90A-1859-B4EE-88D2986D3B02	71	1	19	19
0466A42E-47CF-8D24-FD01-C0B689713128	47	97	43	43
063FBA79-99EC-66FB-29F7-25726D1764A5	47	67	31	5
0668296C-CD89-A883-76BC-2E4C44F8C8AE	89	83	79	79
06CD9AA5-9B42-D684-DDDD-A5E394FEB9A9	43	31	31	31

temp\_orders 36 x

Output

Action Output

#	Time	Action	Message
158	18:10:10	INSERT INTO temp_orders (id, product_id1, product_id2, product_id3, product_id4) SELECT id, CAST(SUB...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
159	18:10:15	SELECT * FROM temp_orders LIMIT 0, 1000	587 row(s) returned

Una vez resuelta la separación de los valores, se deben traspasar todo a una mismo campo respetando sus ids, por lo que creamos una tabla auxiliar a la que denominamos trasp\_order para traspasar los valores desde la tabla temp\_orders. Luego eliminaremos la tabla orders para poder renombrar la tabla trasp\_order como orders. Y finalmente eliminaremos también la tabla temporal.

### Creamos la tabla de traspaso:

```

CREATE TABLE trasp_order (
    id VARCHAR(100),
    product_id INT);

```

### Insertamos los valores desde la tabla temp\_orders:

Para insertar los datos, normalizamos la tabla de multiples productos en diferentes columnas (product\_id1, product\_id2, product\_id3, product\_id4) y los convertimos en filas individuales dentro de la tabla trasp\_order. Con el método UNION ALL combinamos las diferentes consultas.

```

INSERT INTO trasp_order (id, product_id)
SELECT id, product_id1
FROM temp_orders
WHERE product_id1 IS NOT NULL
UNION ALL
SELECT id, product_id2
FROM temp_orders
WHERE product_id2 IS NOT NULL
UNION ALL
SELECT id, product_id3
FROM temp_orders
WHERE product_id3 IS NOT NULL

```

```

UNION ALL
SELECT id, product_id4
FROM temp_orders
WHERE product_id4 IS NOT NULL;

```

```

275 • CREATE TABLE trasp_order (
276     id VARCHAR(100),
277     product_id INT);
278
279 -- Insertamos los valores:
280 • INSERT INTO trasp_order (id, product_id)
281     SELECT id, product_id1
282     FROM temp_orders
283     WHERE product_id1 IS NOT NULL
284     UNION ALL
285     SELECT id, product_id2
286     FROM temp_orders
287     WHERE product_id2 IS NOT NULL
288     UNION ALL
289     SELECT id, product_id3
290     FROM temp_orders
291     WHERE product_id3 IS NOT NULL
292     UNION ALL
293     SELECT id, product_id4
294     FROM temp_orders
295     WHERE product_id4 IS NOT NULL;
296

```

Output

#	Time	Action	Message
161	18:16:02	CREATE TABLE trasp_order (id VARCHAR(100), product_id INT)	0 row(s) affected
162	18:16:26	INSERT INTO trasp_order (id, product_id) SELECT id, product_id1 FROM temp_orders WHERE product_id1 ...	2348 row(s) affected Records: 2348 Duplicates: 0 Warnings: 0

**Comprobamos:**  
SELECT \* FROM trasp\_order;

```

297 • SELECT * FROM trasp_order;

```

Result Grid

	id	product_id
▶	02C6201E-D90A-1859-B4EE-88D2986D3B02	71
	0466A42E-47CF-8D24-FD01-C0B689713128	47
	063FBA79-99EC-66FB-29F7-25726D1764A5	47
	0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89
	06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43

trasp\_order 37 x

Output

#	Time	Action	Message
✓ 162	18:16:26	INSERT INTO trasp_order (id, product_id) SELECT id, product_id1 FROM temp_orders WHERE product_id1 ...	2348 row(s) affected F
✓ 163	18:17:04	SELECT * FROM trasp_order LIMIT 0, 1000	1000 row(s) returned

**Eliminamos la tabla original "orders" la cual será reemplazada con "trasp\_order"**  
DROP table orders;

**Renombrar la tabla "trasp\_order" a su nombre final "orders".**  
ALTER TABLE trasp\_order RENAME orders;

**Finalmente eliminamos la tabla temporal "temp\_orders"**  
DROP TABLE temp\_orders;

```

299 -- Borrar la tabla original "orders" la cual será reemplazada con "trasp_order"
300 • DROP table orders;
301
302 -- Renombrar la tabla "trasp_order" a su valor final "orders".
303 • ALTER TABLE trasp_order RENAME orders;
304
305 -- Borrar tabla temporal "temp_orders"
306 • DROP TABLE temp_orders;
307

```

Output				
Action Output				
#	Time	Action	Message	
✓ 163	18:17:04	SELECT * FROM trasp_order LIMIT 0, 1000	1000 row(s) returned	
✓ 164	18:17:57	DROP table orders	0 row(s) affected	
✓ 165	18:18:03	ALTER TABLE trasp_order RENAME orders	0 row(s) affected	
✓ 166	18:18:09	DROP TABLE temp_orders	0 row(s) affected	

**Comprobamos cómo han quedado las tablas:**

```

309 • SHOW TABLES FROM sales;
310

```

Tables_in_sales
companies
credit_cards
credit_status
orders
products
transactions
users

Result 39 x

Mostramos cómo ha quedado

SELECT \* FROM orders ORDER BY id;

```

311 • SELECT * FROM orders ORDER BY id;
312
313 -- Le cambiamos el nombre a la columna, la vaciamos

```

id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
0466A42E-47CF-8D24-FD01-C0B689713128	47
0466A42E-47CF-8D24-FD01-C0B689713128	43
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	43

orders 40 x

**Finalmente añadimos los Foreign Keys:**

Creamos un índice de id en orders para vincular con la Foreign Key de la tabla transactions y el Primary Key de la tabla products.

ALTER TABLE orders

ADD CONSTRAINT fk\_orders\_products FOREIGN KEY (product\_id) REFERENCES products (id),

ADD CONSTRAINT fk\_orders\_transactions FOREIGN KEY (id) REFERENCES transactions (id);

## Comprobamos:

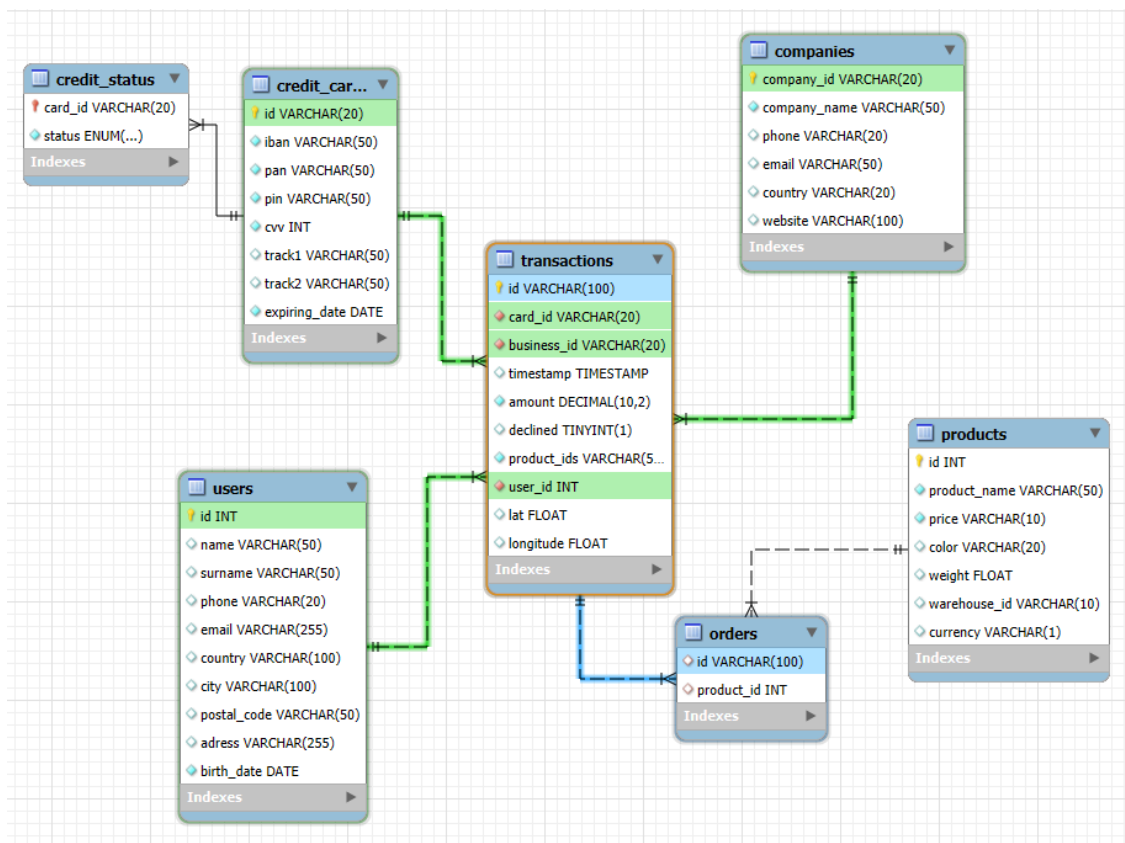
SHOW INDEXES FROM orders;

```
315 -- Creamos un índice de id en orders para vincular con la Foreign Key de la tabla transactions.
316 • ALTER TABLE orders
317 ADD CONSTRAINT fk_orders_products FOREIGN KEY (product_id) REFERENCES products (id),
318 ADD CONSTRAINT fk_orders_transactions FOREIGN KEY (id) REFERENCES transactions (id);
319
320 -- Comprobamos
321 • SHOW INDEXES FROM orders;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Vis
orders	1	fk_orders_products	1	product_id	A	26			YES	BTREE			YES
orders	1	fk_orders_transactions	1	id	A	587			YES	BTREE			YES

#	Time	Action	Message
171	18:24:16	ALTER TABLE orders ADD CONSTRAINT fk_orders_products FOREIGN KEY (product_id) REFERENCES products (id),	2348 row(s) affected Records: 2348 Duplicates: 0 Warnings: 0
172	18:24:21	SHOW INDEXES FROM orders	2 row(s) returned

## ESQUEMA FINAL:



## OPCIÓN 2 más sencilla:

```
CREATE TABLE IF NOT EXISTS orders2 (
    transaction_id VARCHAR(100) NOT NULL,
    product_id INT NOT NULL);
```

```

INSERT INTO orders2 (transaction_id, product_id)
SELECT transactions.id as transaction_id, products.id AS product_id
FROM transactions
JOIN products ON FIND_IN_SET(products.id, REPLACE (transactions.product_ids, " ", "")) > 0;

SELECT * FROM orders2;

```

```
--
29 • INSERT INTO orders2 (transaction_id, product_id)
30 SELECT transactions.id as transaction_id, products.id AS product_id
31 FROM transactions
32 JOIN products ON FIND_IN_SET(products.id, REPLACE (transactions.product_ids, " ", "")) > 0;
33
34 • SELECT * FROM orders2;
```

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3802	71
02C6201E-D90A-1859-B4EE-88D2986D3802	19
02C6201E-D90A-1859-B4EE-88D2986D3802	1
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	47

orders2 1 x

Output

#	Time	Action	Message
193	19:04:36	INSERT INTO orders2 (transaction_id, product_id) SELECT transactions.id as transaction_id, products.id AS product_id FROM ...	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
194	19:04:57	SELECT * FROM orders2 LIMIT 0, 1000	1000 row(s) returned

## Exercici 1

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

```

SELECT product_id, count(orders.product_id) AS cantidad_ventas
FROM transactions
INNER JOIN orders ON transactions.id = orders.id
WHERE transactions.declined = 0
GROUP BY product_id
ORDER BY cantidad_ventas DESC;

```

```
325 • SELECT product_id, count(orders.product_id) AS cantidad_ventas
326 FROM transactions
327 INNER JOIN orders ON transactions.id = orders.id
328 WHERE transactions.declined = 0
329 GROUP BY product_id
330 ORDER BY cantidad_ventas DESC;
```

product_id	cantidad_ventas
67	96
43	95
23	94
2	92
53	89

Result 46 x

Output

#	Time	Action	Message
183	18:50:21	SHOW INDEXES FROM orders	2 row(s) returned
184	18:50:25	SELECT product_id, count(orders.product_id) AS cantidad_ventas FROM transactions INNER JOIN orders ON transactions.id = ...	26 row(s) returned



