

Sprint 4

NIVEL 1 ★

EXTRA | Creación de la base de datos “business_db” (La tabla products se introduce en el lvl3)

- Se crea la base de datos y se selecciona para usar.

```
-- Creamos la database
```

```
create database if not exists business_db default character set utf8mb4;
use business_db;
```

- Tabla “companies”:

- Se crea una tabla temporal para alojar los datos del .csv “companies”, ya que vienen en una única celda como bloque de texto separado por comas.

```
6      -- creamos una tabla "temporal" para los datos raw de companies
7 • create table if not exists companies_raw (
8      companies_raw_data text
9  );
```

- Se introducen los datos en la tabla

```
11     -- introducimos los datos raw de companies
12 • load data
13     infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv"
14     into table companies_raw;
```

- Se crea la tabla definitiva de “companies”. Datos en Varchar para soportar cualquier problema que pueda surgir de formato.

```
18     -- Creamos la tabla final de companies
19 • create table if not exists companies (
20     company_id varchar(15) primary key,
21     company_name varchar(255),
22     phone varchar(15),
23     email varchar(100),
24     country varchar(100),
25     website varchar(255)
26 );
```

- Se separan los datos del archivo .csv y se alojan en sus correspondientes columnas. Para esto se usa el método substring_index() y se utiliza la coma (“,”) como argumento separador.

```
28     -- Separamos la raw data en las columnas de la tabla definitiva
29 • insert into companies (company_id, company_name, phone, email, country, website)
30     select
31         substring_index(companies_raw_data, ",", 1) as company_id,
32         substring_index(substring_index(companies_raw_data, ",", 2), ",", -1) as company_name,
33         substring_index(substring_index(companies_raw_data, ",", 3), ",", -1) as phone,
34         substring_index(substring_index(companies_raw_data, ",", 4), ",", -1) as email,
35         substring_index(substring_index(companies_raw_data, ",", 5), ",", -1) as country,
36         substring_index(substring_index(companies_raw_data, ",", 6), ",", -1) as website
37     from companies_raw;
```

- Se eliminan las cabeceras.

```
40 • delete from companies
41   where company_id = "company_id"; -- Eliminamos cabeceras
```

- Se elimina la tabla temporal.

```
45   -- Eliminamos la tabla temporal
46 • drop table companies_raw;
```

- RESULTADO:

```
43 • select * from companies; -- Check
44
```

Result Grid						
Filter Rows:						
Edit: Export/Import: Wrap Cell Content:						
company_id	company_name	phone	email	country	website	
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://instagram.com/site	
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/group/9	
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/sub/cars	
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.co.uk	Germany	https://cnn.com/user/110	
b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.proin@outlook.ca	New Zealand	https://netflix.com/settings	
b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.co.uk	Norway	https://nytimes.com/user/110	
b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cnn.com/opa	

companies 25 x

Output

Action Output

#	Time	Action	Message
1	11:30:09	select * from companies LIMIT 0, 1000	100 row(s) returned

- Tabla "credit_cards":

- Se crea una tabla temporal para alojar los datos "raw" del .csv "credit_cards".

```
51   -- creamos una tabla "temporal" para los datos raw de credit_card
52 • create table if not exists credit_card_raw (
53   cc_raw_data text
54 );
```

- Se introducen los datos en la tabla.

```
56   -- Introducimos los datos raw de credit_cards
57 • load data
58   infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\credit_cards.csv"
59   into table credit_card_raw;
```

- Se crea la tabla definitiva de "credit_cards".

```
64 • create table if not exists credit_cards (
65   id varchar(15) primary key,
66   user_id varchar(15),
67   iban varchar(50),
68   pan varchar(100),
69   pin varchar(4),
70   cvv varchar(4),
71   track1 varchar(255),
72   track2 varchar(255),
73   expiring_date varchar(20)
74 );
```

- Se separan los datos del archivo .csv y se alojan en sus correspondientes columnas, utilizando el método `substring_index()`.

```

76 -- Separamos la raw data en las columnas de la tabla definitiva
77 • insert into credit_cards (id, user_id, iban, pan, pin, cvv, track1, track2, expiring_date)
78 select
79     substring_index(cc_raw_data, ",", 1) as id,
80     substring_index(substring_index(cc_raw_data, ",", 2), ",", -1) as user_id,
81     substring_index(substring_index(cc_raw_data, ",", 3), ",", -1) as iban,
82     substring_index(substring_index(cc_raw_data, ",", 4), ",", -1) as pan,
83     substring_index(substring_index(cc_raw_data, ",", 5), ",", -1) as pin,
84     substring_index(substring_index(cc_raw_data, ",", 6), ",", -1) as cvv,
85     substring_index(substring_index(cc_raw_data, ",", 7), ",", -1) as track1,
86     substring_index(substring_index(cc_raw_data, ",", 8), ",", -1) as track2,
87     substring_index(substring_index(cc_raw_data, ",", 9), ",", -1) as expiring_date
88 from credit_card_raw;

```

- Se eliminan cabeceras.

```

90 -- Limpiamos la data
91 • delete from credit_cards
92 where id = "id"; -- Eliminamos cabeceras

```

- Se cambia el tipo de data en la columna "id".

```

94 • alter table credit_cards
95 modify user_id smallint; -- Cambiamos el tipo de user id para poder ordenar por el

```

- Para poder trabajar con las "expiring_dates" se han de poder transformar en formato date. Este es el proceso seguido:

- Se crea una nueva columna con el formato deseado (date).

```

97 • alter table credit_cards
98 add column expiring_date_format date;

```

- Se transforman las fechas con el método `str_to_date()` y se introducen en la nueva columna.

```

99 • update credit_cards
100 set expiring_date_format = str_to_date(expiring_date, "%m/%d/%y");

```

- Se elimina la primera columna (la de los datos sin formatear).

```

101 • alter table credit_cards
102 drop column expiring_date;

```

- Se renombra la nueva columna con el nombre de la primera.

```

103 • alter table credit_cards
104 change expiring_date_format expiring_date date;

```

- Se elimina la tabla temporal.

```

108 -- Eliminamos la tabla temporal
109 • drop table credit_card_raw;

```

- RESULTADO:

```

106 • select * from credit_cards; -- Check
107

```

Result Grid								
Filter Rows:								
Edit: Export/Import: Wrap Cell Content: Apply								
	id	user_id	iban	pan	pin	cvv	track1	track2
▶	CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%88383712448554646^WovsxeJpWiev^8604...	%87653863056044187=800716333673
	CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%84621311609958661^UftuyfsSeimxn^06106...	%84149568437843501=510714033071
	CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%82183285104307501^CddytcUxwfdq^5907...	%86778580257827162=690685974007
	CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%87281111956795320^XocddjBkecd^09016...	%84246154489281853=280522391678
	CcU-2966	271	BG72LKTQ15627628377363	448566 886747 7265	4900	130	%84728932322756223^JhlgvsuFbmwig^7202...	%82318571115599881=890821578475
	CcU-2973	270	PT878R6778175097479456746	544 58654 54747 784	8760	887	%84761405753775637^HinninnRleir^7108515	%87816169831446746=1310777729

credit_cards 26 x

Output

Action Output

#	Time	Action	Message
1	11:49:32	select * from credit_cards LIMIT 0, 1000	275 row(s) returned

- Tabla “users”:

- Se crea una tabla temporal para alojar los datos “raw” del .csv “credit_cards”.

```
116      -- creamos una tabla "temporal" para los datos raw de users
117 • create table if not exists users_raw (
118     users_raw_data text
119 );
```

- Se introducen los datos de los archivos .csv “users_usa”, “users_uk” y “users_ca”.

```
121      -- Introducimos los datos raw de users
122 • load data
123 infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_usa.csv"
124 into table users_raw;
125
126 • load data
127 infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_uk.csv"
128 into table users_raw;
129
130 • load data
131 infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_ca.csv"
132 into table users_raw;
```

- Se crea la tabla definitiva de “users”.

```
136      -- Creamos la tabla users
137 • create table if not exists users (
138     id varchar(50),
139     name varchar(100),
140     surname varchar(100),
141     phone varchar(20),
142     email varchar(150),
143     birth_date varchar(100),
144     country varchar(100),
145     city varchar(150),
146     postal_code varchar(100),
147     address varchar(255)
148 );
```

- Se separan los datos del archivo .csv y se alojan en sus correspondientes columnas, utilizando el método substring_index(). En este caso, las fechas vienen separadas por “,”, por lo que hay que filtrarlas por otro separador (“”).

```
150      -- Separamos la raw data en las columnas de la tabla definitiva
151 • insert into users (id, name, surname, phone, email, birth_date, country, city, postal_code, address)
152 select
153     substring_index(users_raw_data, ",", 1) as id,
154     substring_index(substring_index(users_raw_data, ",", 2), ",", -1) as name,
155     substring_index(substring_index(users_raw_data, ",", 3), ",", -1) as surname,
156     substring_index(substring_index(users_raw_data, ",", 4), ",", -1) as phone,
157     substring_index(substring_index(users_raw_data, ",", 5), ",", -1) as email,
158     substring_index(substring_index(users_raw_data, '"', 2), '"', -1) as birth_date,
159     substring_index(substring_index(users_raw_data, ",", 8), ",", -1) as country,
160     substring_index(substring_index(users_raw_data, ",", 9), ",", -1) as city,
161     substring_index(substring_index(users_raw_data, ",", 10), ",", -1) as postal_code,
162     substring_index(substring_index(users_raw_data, ",", 11), ",", -1) as address
163 from users_raw;
```

- Se eliminan las cabeceras.

```
167 • delete from users
168 where name = "name"; -- Eliminamos cabeceras
```

- Se cambia el tipo de data de “id” y se setea como primary key.

```
170 • alter table users
171 modify id smallint primary key; -- Seteamos el tipo correcto para el id
```

- Para poder trabajar con las “birth_dates” se han de poder transformar en formato date. Este es el proceso seguido:

- Se crea una nueva columna con el formato deseado (date).

```
173 • alter table users
174     add column birth_date_format date;
```

- Se transforman las fechas con el método str_to_date() y se introducen en la nueva columna.

```
175 • update users
176     set birth_date_format = str_to_date(birth_date, "%b %d, %Y");
```

- Se elimina la primera columna (la de los datos sin formatear).

```
177 • alter table users
178     drop column birth_date;
```

- Se renombra la nueva columna con el nombre de la primera.

```
179 • alter table users
180     change birth_date_format birth_date date;
```

- Se elimina la tabla temporal

```
184     -- Eliminamos la tabla temporal
185 • drop table users_raw;
```

- RESULTADO:

```
182 • select * from users; -- Check
183
```

Result Grid										
Filter Rows:										
Edit: Export/Import: Wrap Cell Content: 1A										
	id	name	surname	phone	email	country	city	postal_code	address	birth_date
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	United States	Lowell	73544	348-7818 Sagittis St.	1985-11-17
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	United States	Des Moines	59464	903 Sit Ave	1992-08-23
	3	Claran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	United States	Columbus	56518	736-2063 Tellus St.	1998-04-29
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	United States	Kailua	77417	Ap #545-2244 Erat. Rd.	1989-02-18
	5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	United States	Sandy	31564	341-2821 Ultrices Av.	1998-09-26
	6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	United States	Nashville	96838	888-2799 Amet Street	1989-10-15
	7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	United States	Hillsboro	29874	8627 Malesuada Rd.	1981-12-04
	8	Nissim	Franks	(692) 157-3469	egestas.aliquam.fringilla@google.ca	United States	Jackson	61750	Ap #251-7144 Integer St.	1993-08-01

users 1 ×

Output

Action Output

#	Time	Action	Message
1	10:43:29	select * from users LIMIT 0, 1000	275 row(s) returned

- Tabla “transactions”:

- En este caso los datos están bien distribuidos en celdas en el propio documento .csv, por lo que no es necesario alojarlos primero en una tabla temporal. Se crea la tabla definitiva.

```
191     -- Creamos la tabla transactions
192 • create table if not exists transactions (
193     id varchar(255),
194     card_id varchar(15),
195     business_id varchar(15),
196     timestamp varchar(255),
197     amount varchar(100),
198     declined varchar(15),
199     product_ids varchar(255),
200     user_id varchar(50),
201     lat varchar(100),
202     longitude varchar(100) null
203 );
```

- Se introduce la data. Los archivos .csv en español separan sus celdas con “;”. Como el método “load data infile”, asume que por defecto están separadas con “,” se debe especificar que no es así.

```
205      -- Insetamos la data
206 •   load data
207     infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\transactions.csv"
208     into table transactions
209     fields terminated by ";"; -- por defecto lee ","
```

- Se eliminan cabeceras.

```
212 •   delete from transactions
213     where id = "id"; -- Eliminamos cabeceras
```

- Se setea la Primary Key.

```
216 •   alter table transactions
217     modify id varchar(255) primary key; -- Seteamos la PK
```

- Se formatean las fechas como “timestamp”.

```
219 •   alter table transactions
220     modify timestamp timestamp; -- Formateamos fechas
```

- Se formatean los amount como “decimal” (2 decimales).

```
222 •   alter table transactions
223     modify amount decimal(10, 2); -- Formateamos amounts
```

- Se formatea “declined” como “boolean”.

```
225 •   alter table transactions
226     modify declined boolean; -- Formateamos declined
```

- Se formatea “user_id” como “smallint” para poder ordenar por dicho campo y que coincida con la pk de la tabla users.

```
228 •   alter table transactions
229     modify user_id smallint; -- Formateamos user_id para que coincida con la tabla users
```

- Se formatea “lat” como “float”.

```
231 •   alter table transactions
232     modify lat float; -- Formateamos lat
```

- Se formatea “longitude” como “float”.

```
234 •   alter table transactions
235     modify longitude float; -- Formateamos longitude
```

- Se añade la constraint con “credit_cards” (Foreign Key).

```
238 •   alter table transactions
239     add constraint cards_constraint
240     foreign key (card_id) references credit_cards(id); -- Unimos con credit_cards
```

- Se añade la constraint con “companies” (Foreign Key).

```
242 •   alter table transactions
243     add constraint companies_constraint
244     foreign key (business_id) references companies(company_id); -- Unimos con companies
```

- Se añade la constraint con “users” (Foreign Key).

```
246 •   alter table transactions
247     add constraint users_constraint
248     foreign key (user_id) references users(id); -- Unimos con users
```

○ RESULTADO:

```
250 • select * from transactions; -- Check
```

251

Result Grid										
Filter Rows:										
Edit: Export/Import: Wrap Cell Contents: 1A										
id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude	
02C6201E-D90A-1859-B4EE-88D2986D3802	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92	81.9185	-12.5276	
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.9695	-117.525	
063FBA79-99EC-66FB-29F7-25726D1764A5	CcU-2987	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.2227	-129.05	
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	CcU-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265	-34.3593	-100.556	
06CD9AA5-9B42-D684-DDDD-A5E394FEB499	CcU-2959	b-2346	2021-10-26 23:00:01	279.93	0	43, 31	92	33.7381	158.298	
07A46D48-31A3-7E87-65B9-0DA902AD109F	CcU-3225	b-2386	2021-06-28 21:11:42	340.87	1	47, 23	272	38.8342	92.1905	
09DE92CE-6F27-28B7-13B5-9385B2B3B8E2	CcU-3071	b-2298	2021-05-11 20:40:06	303.05	1	67, 7	275	71.1706	10.5757	
0A476ED9-0C13-1962-F87B-D3563924B539	CcU-4359	b-2302	2022-02-26 20:33:54	430.49	0	29, 41, 11	221	-56.4901	114.801	

transactions 3 x

Output

Action Output

#	Time	Action	Message
1	11:08:00	select * from transactions LIMIT 0, 1000	587 row(s) returned

Ejercicio 1 | Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.

Query:

```
257 • use business_db;
258 • select users.id, -- en caso de usuarios con el mismo nombre y apellido
259     users.name,
260     users.surname,
261     count(transactions.id) as transacciones_validas_realizadas
262     from users
263     join transactions
264     on users.id = transactions.user_id
265     where transactions.declined = 0 -- Eliminamos transacciones rechazadas
266     group by users.id
267     having transacciones_validas_realizadas > 30;
```

Resultado:

id	name	surname	transacciones_validas_realizadas
92	Lynn	Riddle	39
267	Ocean	Nelson	39
272	Hedwig	Gilbert	38

Result 4 x

Output

Action Output

#	Time	Action	Message
1	11:11:34	select users.id, -- en caso de usuarios con el mismo nombre y apellido users.name, users.surname, count(tran...	3 row(s) returned

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

Query:

```
270 • use business_db;
271 • select credit_cards.iban,
272 round(avg(transactions.amount), 2) as media_de_amount
273 from credit_cards
274 join transactions
275 on credit_cards.id = transactions.card_id
276 join companies
277 on transactions.business_id = companies.company_id
278 where companies.company_name = "Donec Ltd" and transactions.declined = 0 -- Eliminamos transacciones rechazadas
279 group by credit_cards.id;
```

Resultado:

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	iban	media_de_amount
	PT87806228135092429456346	42.82

Result 6

Output

Action Output

#	Time	Action	Message
✓ 1	11:14:22	select credit_cards.iban, round(avg(transactions.amount), 2) as media_de_amount from credit_cards join trans...	1 row(s) returned

NIVEL 2 ★★

EXTRA | 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:

- Se crea la taula. Contiene 2 columnas: "card_id", que actúa como conector con la tabla "credit_cards"; "tarjeta_estado", que indica si la tarjeta en cuestión está operativa o no.

```
286 -- Creamos la tabla
287 • use business_db;
288 • create table if not exists cards_status (
289     card_id varchar(15) primary key,
290     tarjeta_estado varchar(15),
291     foreign key (card_id) references credit_cards(id)
292 );
```

- Se crean los datos y se introducen en la tabla.

```
295 • use business_db;
296 • insert into cards_status (card_id, tarjeta_estado)
297     select
298         data_estado_tarjeta.card_id,
299         data_estado_tarjeta.tarjeta_estado
300     from
301         (with recent_transactions as (
302             select credit_cards.id,
303                 transactions.declined,
304                 row_number() over (partition by credit_cards.id order by transactions.timestamp desc) as recent_transaction_order
305             from credit_cards
306             join transactions
307             on credit_cards.id = transactions.card_id
308         ),
309         three_latest_transactions as (
310             select recent_transactions.id,
311                 recent_transactions.declined
312             from recent_transactions
313             where recent_transactions.recent_transaction_order < 4
314         ),
315         valid_check as (
316             select three_latest_transactions.id,
317                 count(case when three_latest_transactions.declined = 0 then 1 end) as transacciones_validas,
318                 count(case when three_latest_transactions.declined = 1 then 1 end) as transacciones_no_validas
319             from three_latest_transactions
320             group by three_latest_transactions.id
321         )
322     select valid_check.id as card_id,
323         case when valid_check.transacciones_no_validas = 3 then "CANCELADA" else "OPERATIVA" end as tarjeta_estado
324     from valid_check
325 ) as data_estado_tarjeta;
```

- Contenido de la subquery a introducir
 - “recent_transactions”: Selecciona los id de las tarjetas, si las transacciones asociadas a ellas han sido rechazadas o no y el orden en el dichas transacciones se han realizado en el tiempo por cada tarjeta.

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	id	declined	recent_transaction_order			
▶	CcU-4856	0	1			
	CcU-4849	0	1			
	CcU-4849	0	2			
	CcU-4849	0	3			
	CcU-4849	0	4			
	CcU-4849	0	5			
	CcU-4849	0	6			
	CcU-4849	0	7			
	CcU-4849	0	8			
	CcU-4849	0	9			
	CcU-4849	0	10			
	CcU-4849	0	11			
	CcU-4849	0	12			
	CcU-4849	0	13			
	CcU-4849	0	14			
	CcU-4849	0	15			
	CcU-4849	0	16			
	CcU-4849	0	17			

Result 8 x

Output

Action Output

#	Time	Action	Message
✓ 1	11:27:51	select credit_cards.id, transactions.declined, row_number() over (partition by credit_cards.id order by transacti...	587 row(s) returned

- “three_latest_transactions” selecciona, por cada id de tarjeta, los 3 primeros “recent_transaction_order” de la tabla “recent_transactions” (es decir, donde “recent_transaction_order” sea menor que 4).

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	id	declined			
▶	CcU-2938	0			
	CcU-2938	0			
	CcU-2938	0			
	CcU-2945	0			
	CcU-2945	1			
	CcU-2952	0			
	CcU-2952	1			
	CcU-2959	0			
	CcU-2959	0			
	CcU-2959	0			
	CcU-2966	0			
	CcU-2966	1			

Result 10 x

Output

Action Output

#	Time	Action	Message
✓ 1	11:37:26	with recent_transactions as (select credit_cards.id, transactions.declined, row_number() over (partition by cre...	376 row(s) returned

- “valid_check” agrupa cada id de tarjeta y cuenta cuántas transacciones han sido rechazadas y cuántas no (de las últimas 3).

Result Grid			
Filter Rows:		Export:	Wrap Cell Content:
id	transacciones_validas	transacciones_no_validas	
CcU-2938	3	0	
CcU-2945	1	1	
CcU-2952	1	1	
CcU-2959	3	0	
CcU-2966	1	1	
CcU-2973	1	1	
CcU-2980	1	1	

Result 12 x

Output

Action Output

#	Time	Action	Message
1	11:41:20	with recent_transactions as (select credit_cards.id, transactions.declined, row_number() over (partition by cre...	275 row(s) returned

- Finalmente se establece que tarjetas están operativas y cuáles no (Aquellas cuyas transacciones NO válidas sean 3).

card_id	tarjeta_estado
CcU-2938	OPERATIVA
CcU-2945	OPERATIVA
CcU-2952	OPERATIVA
CcU-2959	OPERATIVA
CcU-2966	OPERATIVA
CcU-2973	OPERATIVA
CcU-2980	OPERATIVA
CcU-2987	OPERATIVA

Result 14 x

Output

Action Output

#	Time	Action	Message
1	11:44:07	with recent_transactions as (select credit_cards.id, transactions.declined, row_number() over (partition by cre...	275 row(s) returned

• RESULTADO:

```
327 • select * from cards_status; -- Check
328
```

Result Grid			
Filter Rows:		Edit:	Export/Import:
		Wrap Cell Content:	
card_id	tarjeta_estado		
CcU-2938	OPERATIVA		
CcU-2945	OPERATIVA		
CcU-2952	OPERATIVA		
CcU-2959	OPERATIVA		
CcU-2966	OPERATIVA		
CcU-2973	OPERATIVA		
CcU-2980	OPERATIVA		
CcU-2987	OPERATIVA		

cards_status 15 x

Output

Action Output

#	Time	Action	Message
1	11:45:31	select * from cards_status LIMIT 0, 1000	275 row(s) returned

Ejercicio 1 | Quantes targetes estan activos?

Query:

```
330 • use business_db;
331 • select cards_status.tarjeta_estado,
332       count(cards_status.card_id) as cantidad
333       from cards_status
334       group by cards_status.tarjeta_estado;
```

Resultado:

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	tarjeta_estado	cantidad			
▶	OPERATIVA	275			

Result 16 x			
Output			
Action Output			
#	Time	Action	Message
✓ 1	11:47:16	select cards_status.tarjeta_estado, count(cards_status.card_id) as cantidad from cards_status group by cards...	1 row(s) returned

NIVEL 3 ★★★★★

EXTRA | 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:

- Taula “products”:

- Se crea una taula temporal para alojar los datos “raw” del .csv “products”.

```
341 -- Creamos una tabla de productos raw
342 • use business_db;
343 • create table if not exists products_raw(
344     products_raw_data text
345 );
```

- Se introducen los datos en la tabla.

```
347 -- Introducimos la data raw
348 • load data
349     infile "C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv"
350     into table products_raw;
```

- Se crea la tabla definitiva de “products”.

```
352 -- Creamos la tabla productos
353 • use business_db;
354 • create table if not exists products(
355     id varchar(5),
356     product_name varchar(255),
357     price varchar(20),
358     colour varchar(50),
359     weight varchar(50),
360     warehouse_id varchar(100)
361 );
```

- Se separan los datos del archivo .csv y se alojan en sus correspondientes columnas, utilizando el método substring_index().

```
363 -- introducimos la data raw en la tabla definitiva
364 • insert into products (id, product_name, price, colour, weight, warehouse_id)
365     select
366         substring_index(products_raw_data, ",", 1) as id,
367         substring_index(substring_index(products_raw_data, ",", 2), ",", -1) as product_name,
368         substring_index(substring_index(products_raw_data, ",", 3), ",", -1) as price,
369         substring_index(substring_index(products_raw_data, ",", 4), ",", -1) as colour,
370         substring_index(substring_index(products_raw_data, ",", 5), ",", -1) as weight,
371         substring_index(substring_index(products_raw_data, ",", 6), ",", -1) as warehouse_id
372     from products_raw;
```

- Se eliminan cabeceras.

```
375 -- Eliminamos cabeceras
376 • delete from products where id = "id";
```

- Se elimina el símbolo de la columna “price” para poder transformar los datos y operar con ellos.

- Se crea una nueva columna.

```
378 -- Creamos una columna nueva
379 • alter table products
380     add column price_dollars decimal(10, 2);
```

- Se elimina el símbolo y se introducen los datos de la primera columna en la nueva.

```
381 -- Filtramos e introducimos en ella el contenido de price
382 • update products
383     set price_dollars = replace(price, "$", "");
```

- Eliminamos la primera columna

```
384          -- Eliminamos la columna original con símbolos
385 •   alter table products
386       drop column price;
```

- Se formatea la columna “id” y se establece como Primary Key:

```
388          -- ID (seteamos pk)
389 •   alter table products
390       modify id smallint primary key;
```

- Se formatea “weight” como float (No se formatea como decimal, porque no se sabe si en un futuro se será más preciso con los gramos, por lo que se deja abierta la posibilidad a colocar los gramos necesarios con los decimales que hagan falta) :

```
391          -- Weight
392 •   alter table products
393       modify weight float;
```

- Se elimina la tabla temporal

```
397       -- Eliminamos la tabla temporal
398 •   drop table products_raw;
```

- RESULTADO:

```
395 •   select * from products; -- Check
396
```

Result Grid						
Filter Rows: <input type="text"/>						
Edit: Export/Import: Wrap Cell Content:						
	id	product_name	colour	weight	warehouse_id	price_dollars
▶	1	Direwolf Stannis	#7c7c7c	1	WH-4	161.11
	2	Tarly Stark	#919191	2	WH-3	9.24
	3	duel tourney Lannister	#d8d8d8	1.5	WH-2	171.13
	4	warden south duel	#111111	3	WH-1	71.89
	5	skywalker ewok	#dbdbdb	3.2	WH-0	171.22

products 18 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:10:32	select * from products LIMIT 0, 1000	100 row(s) returned

- Para poder operar con los “product_id” de la tabla transactions hay que separarlos. Para ello se crea una nueva tabla que contiene el id de la transacción y múltiples columnas con los “product_id” asociados a cada pedido. Esta tabla nunca se unirá con nada, ya que es un paso intermedio, pero necesario.

Tabla “productos_comprados”

- Se crea la tabla.

```
401 •   use business_db;
402 •   create table if not exists productos_comprados(
403       transaction_id varchar(100) primary key,
404       p1 varchar(5),
405       p2 varchar(5),
406       p3 varchar(5),
407       p4 varchar(5)
408   );
```

**Previamente se ha comprobado la cantidad máxima de productos en cada pedido. Si en un futuro se crean transacciones con más cantidad (y se sigue usando este método) habría que editar la tabla para añadir las columnas “px” que sean necesarias.*

- Se introducen los datos de productos pedidos a la nueva tabla desde “transactions”. Para ello se hace una comprobación que revisa cuántos productos hay en cada transacción (hasta un máximo de 4 de momento) y los mete en su columna correspondiente. Si hay menos del máximo de productos, se setean las columnas correspondientes como “null”.

```

410      -- Insertamos la data separada de la columna transactions.product_id
411      use business_db;
412      insert into productos_comprados (transaction_id, p1, p2, p3, p4)
413      select id,
414      substring_index(product_ids, ",", 1) as p1,
415      if (
416          length(product_ids) - length(replace(product_ids, ",", "")) + 1 >= 2,
417          substring_index(substring_index(product_ids, ",", 2), ",", -1),
418          null
419      ) as p2,
420      if (
421          length(product_ids) - length(replace(product_ids, ",", "")) + 1 >= 3,
422          substring_index(substring_index(product_ids, ",", 3), ",", -1),
423          null
424      ) as p3,
425      if (
426          length(product_ids) - length(replace(product_ids, ",", "")) + 1 >= 4,
427          substring_index(substring_index(product_ids, ",", 4), ",", -1),
428          null
429      ) as p4
430      from transactions;

```

- RESULTADO:

```

432      select * from productos_comprados; -- Check

```

Result Grid					
Filter Rows:					
Edit:					
Export/Import:					
Wrap Cell Content:					
transaction_id	p1	p2	p3	p4	
02C6201E-D90A-1859-B4EE-88D2986D3B02	71	1	19	NULL	
0466A42E-47CF-8D24-FD01-C0B689713128	47	97	43	NULL	
063FBA79-99EC-66FB-29F7-25726D1764A5	47	67	31	5	
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89	83	79	NULL	
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43	31	NULL	NULL	

productos_comprados 19 x

Output

Action Output

#	Time	Action	Message
1	12:24:32	select * from productos_comprados LIMIT 0, 1000	587 row(s) returned

- Utilizando como base la tabla “productos_comprados” se crea la tabla “bought_products” que será la que se utilizará realmente en el esquema final (ya que será más sencillo operar con ella y más flexible). Esta tabla consta de dos únicas columnas: “transactions_id” y “product_id”, que se utilizan como Foreign Keys con sus respectivas tablas.

Tabla “bought_products”:

- Se crea la tabla

```
435 • use business_db;
436 • create table if not exists bought_products (
437     transactions_id varchar(255),
438     product_id smallint
439 );
```

- Se introducen los datos de “productos_comprados” en el formato deseado. Se utiliza union de cada cada “id+px”.

```
441 -- Insertamos los datos de productos_comprados en la tabla de bought_products
442 • insert into bought_products (transactions_id, product_id)
443     select transaction_id, p1 from productos_comprados
444     union
445     select transaction_id, p2 from productos_comprados
446     union
447     select transaction_id, p3 from productos_comprados
448     union
449     select transaction_id, p4 from productos_comprados;
```

- Se limpian los registros nulos.

```
451 -- Limpiamos los nulls
452 • delete from bought_products
453     where product_id is null;
```

- Se une la tabla con “transactions”

```
458 -- Union bought_products con transactions
459 • alter table bought_products
460     add constraint transaction_product_fk foreign key (transactions_id) references transactions(id);
```

- Se une la tabla con “products”

```
462 -- Union con bought_products productos
463 • alter table bought_products
464     add constraint bought_product_id_fk foreign key (product_id) references products(id);
```

- RESULTADO:

```
455 • select * from bought_products; -- Check
```

```
456
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	transactions_id	product_id				
▶	02C6201E-D90A-1859-B4EE-88D2986D3802	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				

bought_products20 x

Output:

Action Output

#	Time	Action	Message
✓ 1	12:32:00	select * from bought_products LIMIT 0, 1000	1000 row(s) returned

**Idealmente ahora se eliminarían la tabla “productos_comprados” y la columna transactions.product_id y se utilizaría esta nueva tabla (“bought_products”) para introducir los datos de productos comprados directamente. No obstante, las voy a dejar por un hipotético caso donde sigan llegando registros de esta manera y haya que procesarlos.*

Ejercicio 1 | Necessitem conèixer el nombre de vegades que s'ha venut cada producte.

Query:

```
466 -- Requested query
467 • use business_db;
468 • select products.id,
469     products.product_name,
470     count(bought_products.product_id) as cantidad_comprada
471 from products
472 join bought_products
473 on products.id = bought_products.product_id
474 join transactions
475 on bought_products.transactions_id = transactions.id
476 where transactions.declined = 0
477 group by products.id
478 order by cantidad_comprada desc;
```

Resultado:

	id	product_name	cantidad_comprada
▶	23	riverlands north	60
	67	Winterfell	59
	2	Tarly Stark	56
	43	duel	54
	17	skywalker ewok sith	54
	97	jinn Winterfell	53
	79	Direwolf riverlands the	52
	1	Direwolf Stannis	51

Result 21 x

Output



Action Output

#	Time	Action	Message
✓ 1	12:39:13	select products.id, products.product_name, count(bought_products.product_id) as cantidad_comprada from ...	26 row(s) returned