

DATA ANALYTICS

IT ACADEMY | BCN ACTIVA

SPRINT 04

Creación de base de datos
12 de mayo de 2025



Letiane Benincá
benincalf@gmail.com

TAREA S4.01

CREACIÓN DE BASE DE DATOS

DESCRIPCIÓN

Partiendo de algunos archivos CSV diseñaras y crearas la tuya base de datos.

NIVEL 1

Descarga los archivos CSV, estúdialos y diseña una base de datos con un esquema de estrella que contenga, como mínimo 4 tablas de las cuales puedes realizar las siguientes consultas:

Inicialmente revisamos los datos que tenemos y revisamos los campos de las tablas:

TABLAS				
transaction	companies	credit_cards	user (ca, uk, usa)	products [NIVEL 3]
<i>id</i>	<i>company_id</i>	<i>id</i>	<i>id</i>	<i>id</i>
<i>card_id</i>	<i>company_name</i>	<i>user_id</i>	<i>name</i>	<i>product_name</i>
<i>business_id</i>	<i>phone</i>	<i>iban</i>	<i>surname</i>	<i>price</i>
<i>timestamp</i>	<i>email</i>	<i>pan</i>	<i>phone</i>	<i>colour</i>
<i>amount</i>	<i>country</i>	<i>pin</i>	<i>email</i>	<i>weight</i>
<i>declined</i>	<i>website</i>	<i>cvv</i>	<i>birth_date</i>	<i>warehouse_id</i>
<i>product_ids</i>		<i>track1</i>	<i>country</i>	
<i>user_id</i>		<i>track2</i>	<i>city</i>	
<i>lat</i>		<i>expiring_date</i>	<i>postal_code</i>	
<i>longitude</i>			<i>address</i>	

Tenemos los campos de cada tabla, sus primary keys y foreign keys (en amarillo), para relacionarse entre ellas. Se identifica que los datos de los users son iguales, así que unificamos en solo una tabla. La tabla products, será incorporada para los ejercicios del nivel 3.

Para empezar a crear el database, tenemos que entender como son los datos que serán inseridos, para utilizar la codificación correcta:

* Recomendable inicializar los campos alfanuméricos con VARCHAR(255), para evitar errores en la carga y luego adaptarlos a los tipos de datos que vas a utilizar.

TABLA COMPANY				
COLUMNA	DATOS A INSERIR	TIPOLOGIA	USADO	RELACIÓN
company_id	b-2222	Alfanumérico	VARCHAR(255)	PRIMARY KEY
company_name	Ac Fermentum Incorporated	Alfanumérico	VARCHAR(255)	
phone	06 85 56 52 33	Alfanumérico	VARCHAR(255)	
email	donec.porttitor.tellus@yahoo.net	Alfanumérico	VARCHAR(255)	
country	Germany	Alfanumérico	VARCHAR(255)	
website	https://instagram.com/site	Alfanumérico	VARCHAR(255)	

TABLA CREDIT CARDS				
COLUMNA	DATOS A INSERIR	TIPOLOGIA	USADO	RELACIÓN
<i>id</i>	CcU-2945	8 caracteres alfanuméricos.	VARCHAR(255)	PRIMARY KEY
<i>user_id</i>	274	Entero	INT	FOREING KEY
iban	DO26854763748537475216568689	Alfanumérico	VARCHAR(255)	
pan	5,14242E+15	Alfanumérico	VARCHAR(255)	
pin	9080	Alfanumérico	VARCHAR(255)	
cvv	887	Alfanumérico	VARCHAR(255)	
track1	%B4621311609958661^UftuyfsSeimxn ^0610628241??	Alfanumérico	VARCHAR(255)	
track2	%B4149568437843501=5107140330?1	Alfanumérico	VARCHAR(255)	
expiring_date	08/24/23	Alfanumérico, en este caso no está en patrón de DATE.	VARCHAR(255)	

* En la peer review, comentamos que pin y pan, podrían ser VARCHAR, cambiando de INT, porque si en algún momento hay un número que se inserte con el 0, a principio, no lo tendría en cuenta. Siendo VARCHAR, si mantiene el número (ejemplo: 012).

Para la tabla user, unificamos la información que es igual en los tres excels:

TABLA USERS (ca, uk, usa)				
COLUMNA	DATOS A INSERIR	TIPOLOGIA	USADO	RELACIÓN
<i>id</i>	201	Entero	INT AUTO_INCREMENT	PRIMARY KEY
name	Iola	Alfanumérico	VARCHAR(255)	
surname	Powers	Alfanumérico	VARCHAR(255)	
phone	(0111) 367 0184	Alfanumérico	VARCHAR(255)	
email	ante.blandit@outlook.edu	Alfanumérico	VARCHAR(255)	
birth_date	Mar 20, 2000	Alfanumérico, en este caso no está en patrón de DATE.	VARCHAR(255)	
country	Canada	Alfanumérico	VARCHAR(255)	
city	Rigolet	Alfanumérico	VARCHAR(255)	
postal_code	V6T 6M7	Alfanumérico	VARCHAR(255)	
address	154-5415 Auctor St.	Alfanumérico	VARCHAR(255)	

TABLA TRANSACTIONS				
COLUMNA	DATOS A INSERIR	TIPOLOGIA	USADO	RELACIÓN
<i>id</i>	108B1D1D-5B23-A76C-55EF-C568E49A05DD	36 caracteres alfanuméricos.	VARCHAR(255)	PRIMARY KEY
<i>card_id</i>	CcU-2938	8 caracteres alfanuméricos.	VARCHAR(255)	FOREING KEY
<i>business_id</i>	b-2222	6 caracteres alfanuméricos.	VARCHAR(255)	FOREING KEY
timestamp	07/07/2021 17:43	Almacena data e hora	TIMESTAMP	
amount	293.57	Valores decimales con precisión exacta	DECIMAL (10,2)	
declined	0 ou 1	Valores binarios (0/1, verdadero/falso).	BOOLEAN	
product_ids	11, 13, 61, 29	Almacena múltiples IDs como texto	VARCHAR(255)	
<i>user_id</i>	275	Números enteros	INT	FOREING KEY
lat		Números decimales	FLOAT	
longitude	-178.860.353.536	Números decimales	FLOAT	

Creación del database y tablas. Iniciamos con las tablas de dimensiones:

Tabla db_company (dimensiones):

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. The 'Tables' node under it contains 'db_company'. The central pane displays the SQL code for creating the 'db_company' table:

```

1 -- TAREA S4.01: Creación de base de datos
2 • CREATE DATABASE db_sp04_transaction;
3 • USE db_sp04_transaction;
4
5 -- Creación de las tablas: primero creamos las tablas de dimensiones:
6 -- Tabla company:
7 • CREATE TABLE IF NOT EXISTS db_company (
8     company_id VARCHAR(255) PRIMARY KEY,
9     company_name VARCHAR(255),
10    phone VARCHAR(255),
11    email VARCHAR(255),
12    country VARCHAR(255),
13    website VARCHAR(255)
14 );
15
16

```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
3	10:12:02	USE db_sp04_transaction	0 row(s) affected	0.000 sec
4	10:12:14	CREATE TABLE IF NOT EXISTS db_company (company_id VARCHAR(2...)	0 row(s) affected	0.031 sec

Tabla db_credit card (dimensiones):

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. The 'Tables' node under it contains 'db_credit_card'. The central pane displays the SQL code for creating the 'db_credit_card' table:

```

15
16 -- Creación tabla tarjetas de credito
17 • CREATE TABLE IF NOT EXISTS db_credit_card (
18     id VARCHAR(255) PRIMARY KEY,
19     user_id INT,
20     iban VARCHAR(255),
21     pan VARCHAR(255),
22     pin VARCHAR(255),
23     cvv VARCHAR(255),
24     track1 VARCHAR(255),
25     track2 VARCHAR(255),
26     expiring_date VARCHAR(255)
27 );
28
29
30
31

```

The 'Output' pane at the bottom shows the execution results:

#	Time	Action	Message	Duration / Fetch
4	10:12:14	CREATE TABLE IF NOT EXISTS db_company (company_id VARCHAR(2...)	0 row(s) affected	0.031 sec
5	10:13:47	CREATE TABLE IF NOT EXISTS db_credit_card (id VARCHAR(255) PRI...	0 row(s) affected	0.031 sec

Tabla db_user (dimensiones):

The screenshot shows the MySQL Workbench interface with the 'Information' tab selected. In the left sidebar, under the 'Schemas' section, 'db_sp04_transaction' is expanded to show its tables: db_company, db_credit_card, db_user, db_transactions, db_views, and db_stored_procedures. The main pane displays the SQL code for creating the db_user table:

```

28
29      -- Creación tabla user:
30  • CREATE TABLE IF NOT EXISTS db_user (
31      id INT AUTO_INCREMENT PRIMARY KEY,
32      name VARCHAR(255),
33      surname VARCHAR(255),
34      phone VARCHAR(255),
35      email VARCHAR(255),
36      birth_date VARCHAR(255),
37      country VARCHAR(255),
38      city VARCHAR(255),
39      postal_code VARCHAR(255),
40      address VARCHAR(255)
41  );
42
43
44

```

The 'Output' pane at the bottom shows two actions in the 'Action Output' table:

#	Time	Action	Message	Duration / Fetch
5	10:13:47	CREATE TABLE IF NOT EXISTS db_credit_card (id VARCHAR(255) PRI...)	0 row(s) affected	0.031 sec
6	10:18:39	CREATE TABLE IF NOT EXISTS db_user (id INT AUTO_INCREMENT P...)	0 row(s) affected	0.047 sec

Tablas db_transactions (tabla de hechos):

The screenshot shows the MySQL Workbench interface with the 'Information' tab selected. In the left sidebar, under the 'Schemas' section, 'db_sp04_transaction' is expanded to show its tables: db_company, db_credit_card, db_user, db_transactions, db_views, and db_stored_procedures. The main pane displays the SQL code for creating the db_transactions table:

```

44 • CREATE TABLE IF NOT EXISTS db_transactions (
45     id VARCHAR(255) PRIMARY KEY,
46     card_id VARCHAR(255),
47     business_id VARCHAR(255),
48     timestamp TIMESTAMP,
49     amount DECIMAL(10,2),
50     declined BOOLEAN,
51     product_ids VARCHAR(255),
52     user_id INT,
53     lat FLOAT,
54     longitude FLOAT,
55     FOREIGN KEY (business_id) REFERENCES db_company (company_id),
56     FOREIGN KEY (card_id) REFERENCES db_credit_card (id),
57     FOREIGN KEY (user_id) REFERENCES db_user (id)
58 );
59
60

```

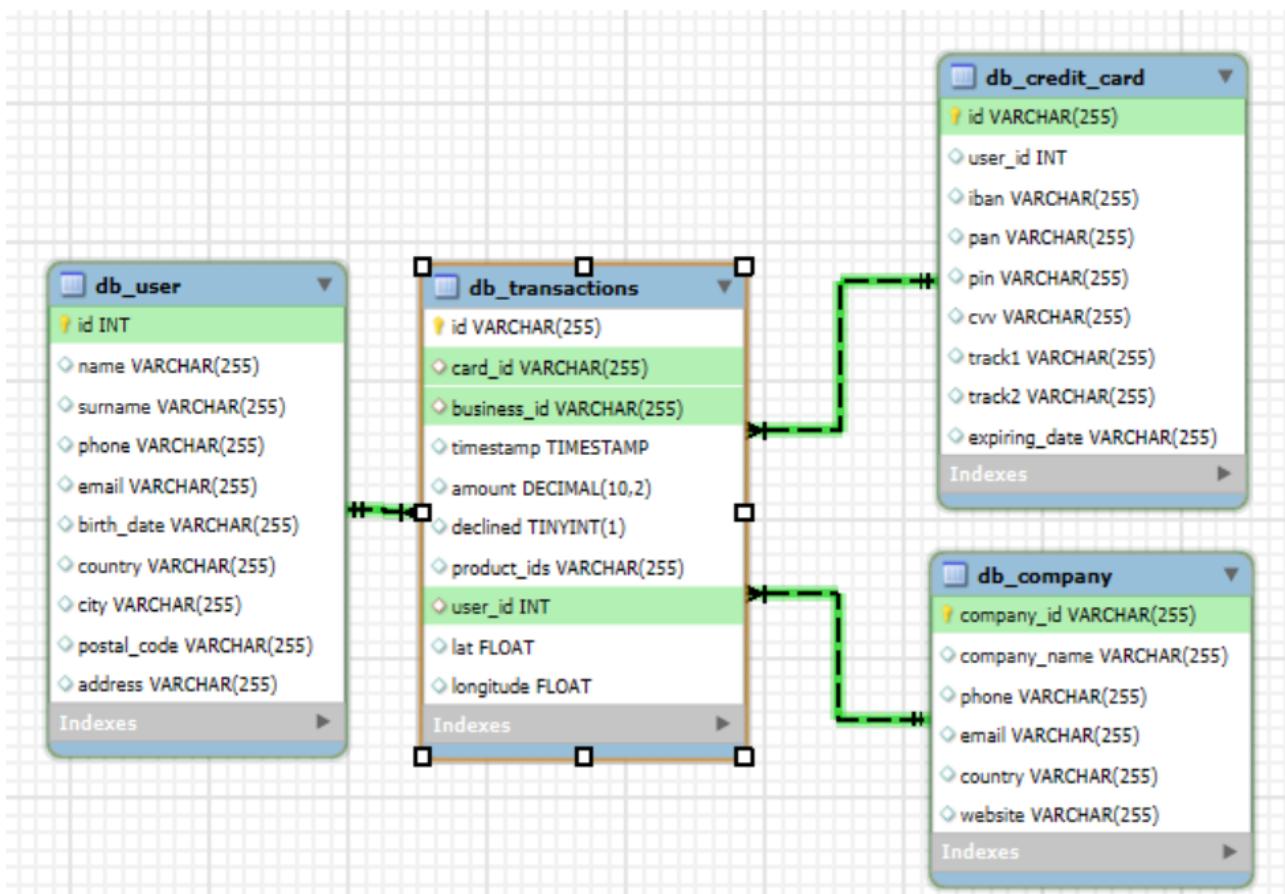
The 'Output' pane at the bottom shows two actions in the 'Action Output' table:

#	Time	Action	Message	Duration / Fetch
6	10:18:39	CREATE TABLE IF NOT EXISTS db_user (id INT AUTO_INCREMENT P...)	0 row(s) affected	0.047 sec
7	10:20:37	CREATE TABLE IF NOT EXISTS db_transactions (id VARCHAR(255) PRI...)	0 row(s) affected	0.078 sec

Se ha creado un modelo en estrella, donde la tabla de hechos es db_transactions (1:N) y las tres dimensiones son:

- *db_credit_card (datos de las tarjetas)*: un usuario puede tener muchas tarjetas de crédito.
- *db_user (datos de los usuarios, combinando tablas de UK, CA y USA)*: un usuario puede realizar muchas transacciones.
- *db_company (datos de las compañías)*: una empresa puede recibir muchas transacciones.

En este sentido, el modelo relacional se presenta de esta forma:



Como segundo paso, inserimos los datos fornecidos:

Primero, verificamos el formato de los datos, si están separados por coma o punto y coma:

Formato datos 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://instagram.com/site
  
```

Formato datos credit_cards:

```

id,user_id,iban,pan,pin,cvv,track1,track2,expiring_date
CcU-2938,275,TR301950312213576817638661,5424465566813633,3257,984,%B8383712448554646^WovsxejDpwiev^86041142?7,%B7653863056044187=8007163336?3,10/30/22
  
```

Formato datos users:

```

id,name,surname,phone,email,birth_date,country,city,postal_code,address
151,Meghan,Hayden,0800 746 6747,arcu.vel@hotmail.ca,"Jul 2, 1980",United Kingdom,Tullibody,A1Y 3TC,Ap #432-4493
Aliquet Rd.
  
```

Formato datos transactions:

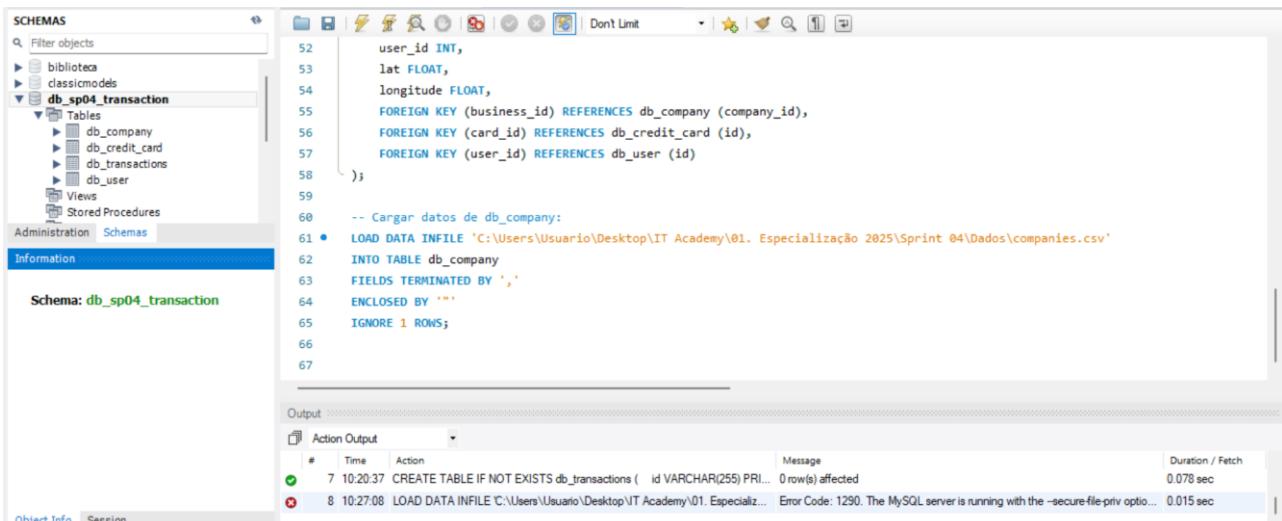
```
id;card_id;business_id;timestamp;amount;declined;product_ids;user_id;lat;longitude
108B1D1D-5B23-A76C-55EF-C568E49A05DD;CcU-2938;b-2222;2021-07-07
17:43:16;293.57;0;59;275;83.7839152128;-178.860353536
7DC26247-20EC-53FE-E555-B6C2E55CA5D5;CcU-2945;b-2226;2022-02-04 15:52:56;312.5;0;71,
41;275;58.9367181312;-76.8171099136
```

Para importar los datos de un Excel, en este caso archivo csv, utilizamos el siguiente comando ([MySQL Manual](#)):

1. LOAD DATA INFILE C:\Users\Usuario\Desktop\IT Academy\01. Especialização 2025\Sprint 04\Dados\ tabla.csv'. --Esta línea carga los datos directamente un archivo csv.
**CUIDADO: Tiene que ser / y no **
2. INTO TABLE **nombre table**--Aquí como en la inserción normal, indicamos el nombre de la tabla donde irán los datos.
3. FIELDS TERMINATED BY ','. – Aquí definimos que los campos están separados (delimitador de campos) por coma (,) para las tablas companies, credit_cards y users y por punto y coma (;) para la tabla transactions.
4. ENCLOSED BY '\"' – Esto explica que los campos pueden estar con doble comillas. Usual utilizarlo cuando estamos cargando datos csv.
5. IGNORE 1 ROWS; -- Por fin, pedimos para que ignore la primera fila, que es nombre de cada columna.

Inicialmente, como primero intento para insertar los datos, salió el error:

Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement:



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is open, showing the 'db_sp04_transaction' schema selected. Under this schema, the 'Tables' node is expanded, showing four tables: db_company, db_credit_card, db_transactions, and db_user. Below the schema tree, the 'Information' panel shows the current schema is 'db_sp04_transaction'. In the center, the SQL editor contains the following code:

```

52     user_id INT,
53     lat FLOAT,
54     longitude FLOAT,
55     FOREIGN KEY (business_id) REFERENCES db_company (company_id),
56     FOREIGN KEY (card_id) REFERENCES db_credit_card (id),
57     FOREIGN KEY (user_id) REFERENCES db_user (id)
58 );
59
60 -- Cargar datos de db_company:
61 • LOAD DATA INFILE 'C:\Users\Usuario\Desktop\IT Academy\01. Especialização 2025\Sprint 04\Dados\companies.csv'
62 INTO TABLE db_company
63 FIELDS TERMINATED BY ','
64 ENCLOSED BY '\"'
65 IGNORE 1 ROWS;
66
67

```

The 'Output' pane at the bottom shows two rows of log information:

#	Time	Action	Message	Duration / Fetch
7	10:20:37	CREATE TABLE IF NOT EXISTS db_transactions (id VARCHAR(255) PRI... 0 row(s) affected		0.078 sec
8	10:27:08	LOAD DATA INFILE 'C:\Users\Usuario\Desktop\IT Academy\01. Especializ... Error Code: 1290. The MySQL server is running with the --secure-file-priv optio...	0.015 sec	

En este caso, se ha buscado el camino default de MySQL para salvar los archivos que serán utilizados:

```

SCHEMAS
  Filter objects
  biblioteca
  classimodels
  db_sp04_transaction
    Tables
      db_company
      db_credit_card
      db_transactions
      db_user
    Views
    Stored Procedures
Administration Schemas
Information
Schema: db_sp04_transaction
Result Grid | Filter Rows: | Export: | Wrap Cell Content: |
Variable_name Value
secure_file_priv C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

Result 1 ×
Output
Action Output
# Time Action Message Duration / Fetch
8 10:27:08 LOAD DATA INFILE 'C:\Users\Usuario\Desktop\IT Academy\01. Especializ...' Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement. 0.015 sec
9 10:27:56 SHOW VARIABLES LIKE secure_file_priv' 1 row(s) returned 0.016 sec / 0.000 sec

Object Info Session
Query Completed
  
```

Inserción de los datos de la tabla db_company:

```

SCHEMAS
  Filter objects
  biblioteca
  classimodels
  db_sp04_transaction
    Tables
      db_company
      db_credit_card
      db_transactions
      db_user
    Views
    Stored Procedures
Administration Schemas
Information
Schema: db_sp04_transaction
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@cloud.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 Taculis Nec Foundation 08 23 04 99 53 sed.dictum.prin@outlook.ca New Zealand https://netfix.com/settings

db_companys 2 ×
Output
Action Output
# Time Action Message Duration / Fetch
10 10:28:49 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv' 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0 0.047 sec
11 10:29:36 SELECT * FROM db_company 100 row(s) returned 0.015 sec / 0.000 sec

Object Info Session
Query Completed
  
```

Inserción de los datos de la tabla db_credit_card:

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. In the main pane, a SQL editor window displays the following code:

```

80 -- Cargar datos de db_credit_card;
81 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
82 INTO TABLE db_credit_card
83 FIELDS TERMINATED BY ','
84 ENCLOSED BY '\"'
85 IGNORE 1 ROWS;
86
87 • SELECT *
88 FROM db_credit_card;

```

Below the code, a 'Result Grid' shows the imported data:

	id	user_id	iban	pan	pin	cvv	track1	track2
▶	CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%B838371244855464~WovsxeJOpiew^8604...	%B7653863056044187=80
	CcU-2945	274	D026854763748537475216568689	514242821948828	9080	887	%B462131609958661~UltuyfsSeimx^06106...	%B4149568437843501=51
	CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%B2183285104307501~CdyyrtcUxwfdq^5907...	%B6778580257821762=69
	CcU-2959	272	CR72427724433841535	372461377349375	3583	667	%B7281111956795320~XocddjBkecd^09016...	%B4246154489281853=28

At the bottom, the 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
12	10:30:42	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0	0.062 sec
13	10:30:53	SELECT * FROM db_credit_card	275 row(s) returned	0.000 sec / 0.000 sec

Inserción de los datos de la tabla db_user:

Aquí, nos presentó el siguiente error: *Error Code: 1262. Row 8 was truncated; it contained more data than there were input columns.*

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. In the main pane, a SQL editor window displays the following code:

```

82 INTO TABLE db_credit_card
83 FIELDS TERMINATED BY ','
84 ENCLOSED BY '\"'
85 IGNORE 1 ROWS;
86
87 • SELECT *
88 FROM db_credit_card;
89
90 -- Cargar datos de db_user:
91 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'
92 INTO TABLE db_user
93 FIELDS TERMINATED BY ','
94 ENCLOSED BY '\"'
95 IGNORE 1 ROWS;
96
97

```

At the bottom, the 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
13	10:30:53	SELECT * FROM db_credit_card	275 row(s) returned	0.000 sec / 0.000 sec
14	10:32:20	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/...	Error Code: 1262. Row 8 was truncated; it contained more data than there w...	0.062 sec

The screenshot shows a Windows Notepad window titled 'users_uk.csv'. The file contains the following data:

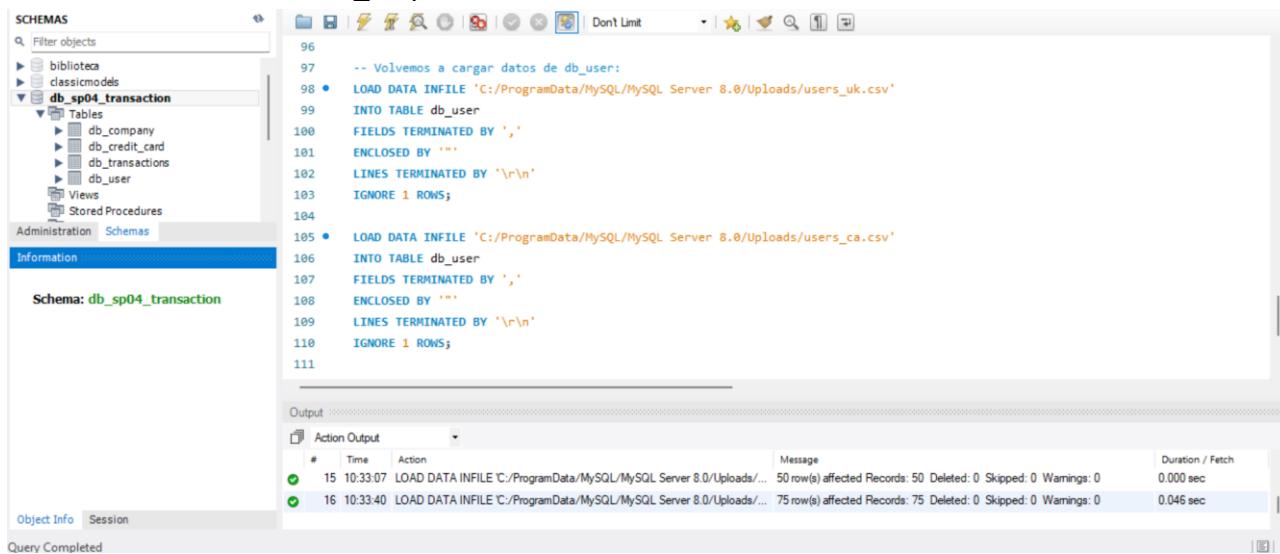
Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9
152,Hakeem,Alford,(0111) 367 0184,adipiscing.ligula@google.edu,"Sep 30, 1979",United Kingdom,Kettering,O21 7JV,551-8930 Lobortis Street								
153,Keegan,Pugh,(016977) 3851,sodales.nisi@aol.org,"Jul 27, 1994",United Kingdom,Whitehaven,HQ8V 7YP,Ap #312-5898 Consectetuer St.								
154,Cooper,Bullock,(021) 2521 6627,et@outlook.net,"Nov 2, 1986",United Kingdom,Presteigne,U18 0DN,872-1866 Pede Rd.								
155,Joshua,Russell,055 4409 5286,justo.nec.ante@outlook.edu,"Jan 23, 1984",United Kingdom,Hatfield,B5H 5CS,Ap #285-4727 Auctor. Av.								
156,Remedios,Case,055 3114 1566,mollis.phasellus.libero@aol.com,"Oct 9, 1994",United Kingdom,North Berwick,QR0 8CW,479-3690 Turpis Road								
157,Philip,Carey,0800 640 6251,phasellus@yahoo.net,"Oct 10, 1992",United Kingdom,Lochgilphead,CE2 6HT,196-1103 Quisque Street								
158,Giovanni,Dove,0000 1111,adipiscing.elit,et,"Dec 24, 1990",United Kingdom,Watlington,TQ9 6YD,Ap #442-7101 Quisque Street, St. "								

The status bar at the bottom indicates 6,350 characters.

El error ocurre cuando una línea del archivo contiene más columnas de lo que esperado por la tabla, o cuando los archivos fueron creados con Windows (en este caso, como en la figura de arriba, Windows (CRLF)). Así, es necesario especificar que la importación de los datos tenga las líneas que terminen con \r\n."

- **LINES TERMINATED BY '\r\n'** – Combinación de retorno de carro + avance de línea.

Inserción de los datos user_uk y ca:

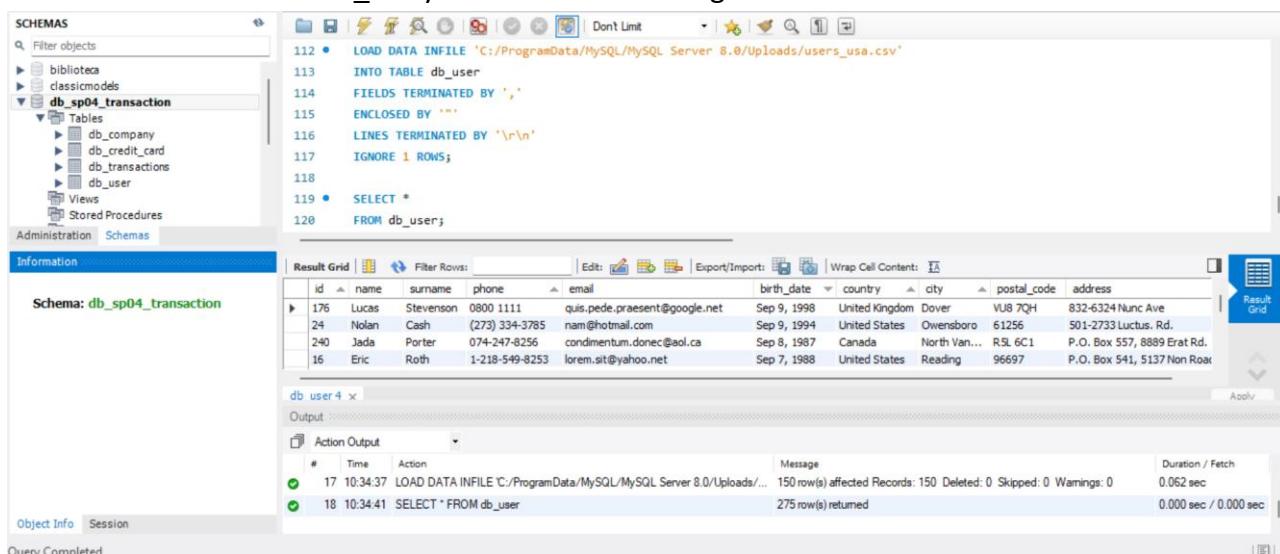


```

96
97  -- Volvemos a cargar datos de db_user:
98 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'
99  INTO TABLE db_user
100 FIELDS TERMINATED BY ','
101 ENCLOSED BY ""
102 LINES TERMINATED BY '\r\n'
103 IGNORE 1 ROWS;
104
105 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv'
106 INTO TABLE db_user
107 FIELDS TERMINATED BY ','
108 ENCLOSED BY ""
109 LINES TERMINATED BY '\r\n'
110 IGNORE 1 ROWS;
111

```

Inserción de los datos user_usa y verificación de la carga:



```

112 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv'
113 INTO TABLE db_user
114 FIELDS TERMINATED BY ','
115 ENCLOSED BY ""
116 LINES TERMINATED BY '\r\n'
117 IGNORE 1 ROWS;
118
119 • SELECT *
120 FROM db_user;

```

ID	Name	Surname	Phone	Email	Birth Date	Country	City	Postal Code	Address
176	Lucas	Stevenson	0800 1111	quis.pede.praesent@google.net	Sep 9, 1998	United Kingdom	Dover	VU8 7QH	832-6324 Nunc Ave
24	Nolan	Cash	(273) 334-3785	nam@hotmail.com	Sep 9, 1994	United States	Owensboro	61256	501-2733 Luctus, Rd.
240	Jada	Porter	074-247-8256	condimentum.donec@aol.ca	Sep 8, 1987	Canada	North Van...	RSL 6C1	P.O. Box 557, 8889 Erat Rd.
16	Eric	Roth	1-218-549-8253	lorem.sit@yahoo.net	Sep 7, 1988	United States	Reading	96697	P.O. Box 541, 5137 Non Roar

Mirando a la columna country, verificamos que los 3 países se han insertado correctamente. Totalizando en 275 usuarios.

Por último, insermos los datos de las db_transactions.

Como los datos del Excel, en esta tabla en específico está delimitado por ';', modificamos el FIELDS TERMINATED BY ',' por FIELDS TERMINATED BY ';':

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree shows the 'db_sp04_transaction' schema selected. In the main pane, a SQL editor window displays the following code:

```

123 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
124   INTO TABLE db_transactions
125   FIELDS TERMINATED BY ';'
126   ENCLOSED BY ""
127   LINES TERMINATED BY '\r\n'
128   IGNORE 1 ROWS;
129
130 • SELECT *
131   FROM db_transactions;
  
```

Below the code, the 'Result Grid' shows the imported data:

	id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
1	02C6201E-D90A-1859-B4EE-8BD298603B02	CdU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92	81.9185	-12.5276
2	0466A42E-47CF-8D24-FD01-C08689713128	CdU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.9695	-117.525
3	063FB479-99EC-6F6B-29F7-2572601764A5	CdU-2987	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.2227	-129.05
4	0668296C-CD89-A883-76BC-2E4C4F8C8AE	CdU-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265	-34.3593	-100.556
5	06CD9AA5-9B42-0684-0DD0-AE394FEBAA99	CdU-2959	b-2346	2021-10-26 23:00:01	279.93	0	43, 31	92	33.7381	158.298

Below the grid, the 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
20	10:52:23	LOAD DATA INFILE C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0	0.140 sec
21	10:52:48	SELECT * FROM db_transactions	587 row(s) returned	0.016 sec / 0.000 sec

Tenemos en total 587 transacciones realizadas.

EJERCICIO 01

Realiza una subconsulta que muestre todos los usuarios con más de 30 transacciones utilizando como mínimo dos tablas.

Para este ejercicio, si utilizó la tabla db_user (campos id, name y surname) y la tabla db_transactions. La subconsulta se realizó en la cláusula **WHERE**, haciendo la combinación de las dos tablas, siendo utilizado el **HAVING** para filtrar los usuarios con más de 30 transacciones.

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tree shows the 'db_sp04_transaction' schema selected. In the main pane, a SQL editor window displays the following code:

```

132
133 -- Nivel 01: Ejercicio 01:
134 -- Realiza una subconsulta que muestre todos los usuarios con más de 30 transacciones utilizando como mínimo dos tablas.
135 • SELECT id, name, surname
136   FROM db_user
137   WHERE id IN (
138     SELECT user_id
139       FROM db_transactions
140       GROUP BY 1
141       HAVING COUNT(id) > 30);
  
```

Below the code, the 'Result Grid' shows the results of the query:

	id	name	surname
1	92	Lynn	Riddle
2	267	Ocean	Nelson
3	272	Hedwig	Gilbert
4	275	Kenyon	Hartman

Below the grid, the 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetch
21	10:52:48	SELECT * FROM db_transactions	587 row(s) returned	0.016 sec / 0.000 sec
22	10:54:04	SELECT id, name, surname FROM db_user WHERE id IN (SELECT user_id ...)	4 row(s) returned	0.000 sec / 0.000 sec

Tenemos a 4 usuarios que hicieron más de 30 transacciones.

EJERCICIO 2

Muestra el promedio del monto por IBAN de las tarjetas de crédito hechas a la compañía Donec Ltd, utiliza como mínimo 2 tablas.

Para esta consulta, se ha hecho una JOIN entre las tres tablas (db_company, db_transactions, db_credit_card) para llegar al resultado esperado. En la consulta, pedimos el AVG del amount y mediante a un filtro WHERE, filtramos la empresa Donec Ltd (en la tabla db_transactions como el business_id = b-2242):

```

SELECT co.company_name,
       cr.iban,
       ROUND(AVG(tr.amount), 2) AS Promedio_ventas
  FROM db_transactions tr
  JOIN db_credit_card cr
    ON tr.card_id = cr.id
  JOIN db_company co
    ON tr.business_id = co.company_id
   WHERE co.company_name = 'Donec Ltd'
  GROUP BY 1,2;
  
```

company_name	iban	Promedio_ventas
Donec Ltd	PT87806228135092429456346	203.72

Result 9 ×

Action Output

#	Time	Action	Message	Duration / Fetch
25	10:56:09	SELECT us.id, us.name, us.surname, CASE WHEN COUNT(tr.id) > 30 THEN ...	216 row(s) returned	0.000 sec / 0.016 sec
26	10:57:16	SELECT co.company_name, cr.iban, ROUND(AVG(tr.amount), 2) AS Prom... 1 row(s) returned		0.016 sec / 0.000 sec

Como resultado tenemos solo un IBAN, que ha realizado dos compras de 354,61 y 42,82, totalizando un promedio de 203,72.

NIVEL 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas y genera la siguiente consulta:

Creamos una tabla db_status_credit_card:

```

178 -- Creación tabla status de las tarjetas de credito
179 • CREATE TABLE IF NOT EXISTS db_status_credit_card (
180     id VARCHAR(255) PRIMARY KEY,
181     status VARCHAR(255),
182     FOREIGN KEY (id) REFERENCES db_credit_card (id)
183 );
184
185 • SELECT *
186   FROM db_status_credit_card;
  
```

Result Grid (empty)

Action Output

#	Time	Action	Message	Duration / Fetch
28	10:59:40	CREATE TABLE IF NOT EXISTS db_status_credit_card (id VARCHAR(2...)	0 row(s) affected	0.047 sec
29	10:59:45	SELECT * FROM db_status_credit_card	0 row(s) returned	0.000 sec / 0.000 sec

Para hacer la creación de los datos de la columna status, fue creada una condición para que pudiera llenar los datos, de acuerdo con el enunciado: *refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas.*

Esta tarea, fue particularmente complicada, y después de muchos intentos, solucionamos con este código (revisado por peer):

```

188 -- Insertar los datos a las tarjetas, con la condición:
189 • INSERT INTO db_status_credit_card (id, status)
190   SELECT
191     card_id,
192     CASE
193       WHEN SUM(declined) >= 3 THEN 'disabled'
194       ELSE 'activated'
195     END AS status
196   FROM (
197     SELECT
198       card_id,
199       declined,
200       ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS transacciones
201     FROM db_transactions
202   ) AS ultimas_3_transacciones
203   WHERE transacciones <= 3 -- Seleccionamos solo las últimas 3 transacciones por tarjeta
204   GROUP BY card_id;
  
```

Action Output

#	Time	Action	Message	Duration / Fetch
29	10:59:45	SELECT * FROM db_status_credit_card	0 row(s) returned	0.000 sec / 0.000 sec
30	11:00:59	INSERT INTO db_status_credit_card (id, status) SELECT card_id, CA...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0	0.031 sec

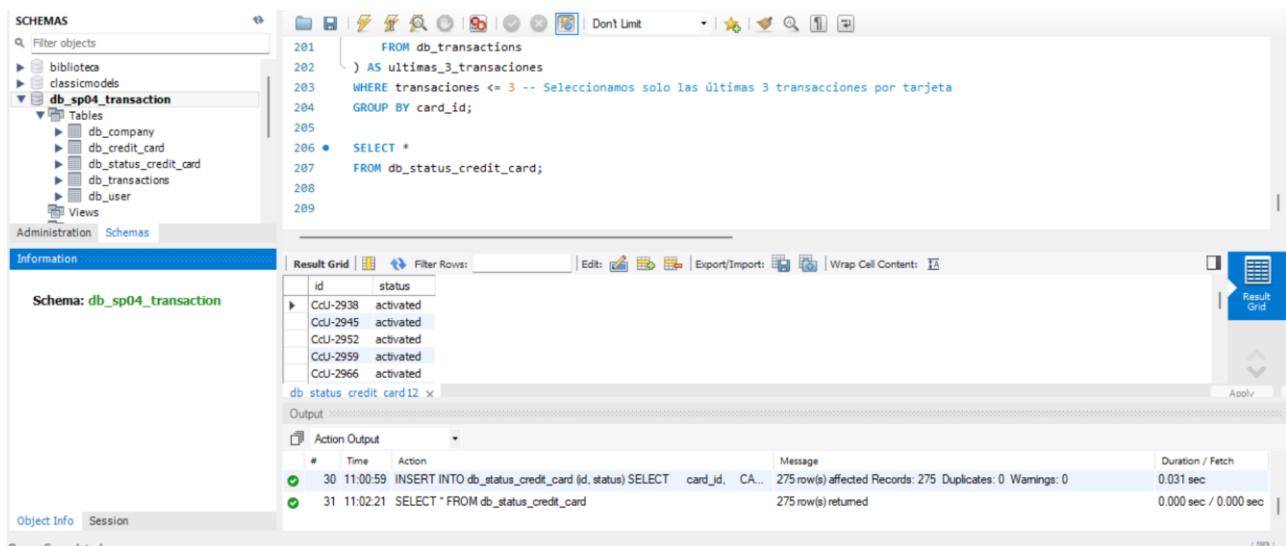
Inicialmente, utilizamos el **INSERT INTO** para insertar los datos en la tabla **db_status_credit_card**, en sus dos columnas: **id** y **status**. Para esto, creamos una condición, relacionado con las transacciones de cada tarjeta. Con el **CASE WHEN** creamos la condición, con el **SELECT** buscamos las informaciones de la tabla **db_transactions**, con una subconsulta que mira cada transacción y organiza de la más reciente de cada tarjeta usando la función **ROW_NUMBER()**. Esto numera las transacciones de cada tarjeta (**PARTITION BY card_id**) empezando de más reciente (**ORDER BY timestamp DESC**).

Después el código filtra solo las tres últimas transacciones de cada tarjeta con la condición **WHERE transacciones <= 3**. Agrupamos por el **card_id**, usando la función **SUM(declined)**, para contar cuantas estas últimas (utilizando el **timestamp**, para verificar las fechas) transacciones fueron declinadas. Se este sumatorio es mayor o igual a 3 el estatus será '**disabled**', si no es así será '**activated**'.

- **ROW_NUMBER()** – asigna um número de fila único.
- **OVER(ORDER BY columna)** – Indica cómo deben ordenarse los datos para asignar los números del **ROW_NUMBER**. (En este caso, ordenamos por **timestamp**, así tenemos las últimas 3 transacciones)
- **PARTITION BY** – divide los datos, para dividir los ids de la columna **product_ids** de **db_transactions**.

Fuentes: [learnsql](#), [stackoverflow](#), [sqlshack](#)

Hecha la creación de la tabla y añadido los campos, verificamos si la tabla tiene la información correcta:



The screenshot shows a database interface with the following details:

- SCHEMAS:** A sidebar showing the schema structure with **biblioteca**, **classicmodels**, and **db_sp04_transaction** selected. Under **db_sp04_transaction**, there are **Tables** (db_company, db_credit_card, db_status_credit_card, db_transactions, db_user) and **Views**.
- Information:** A main pane showing the creation of the **db_status_credit_card** table. The SQL code is as follows:

```

201     FROM db_transactions
202 ) AS ultimas_3_transacciones
203 WHERE transacciones <= 3 -- Seleccionamos solo las últimas 3 transacciones por tarjeta
204 GROUP BY card_id;
205
206 • SELECT *
207   FROM db_status_credit_card;
208
209

```

- Result Grid:** A table showing the results of the query. It has columns **id** and **status**. The data is:

id	status
CclU-2938	activated
CclU-2945	activated
CclU-2952	activated
CclU-2959	activated
CclU-2966	activated
- Action Output:** A log of actions taken during the session. The last two entries are:

#	Time	Action	Message	Duration / Fetch
30	11:00:59	INSERT INTO db_status_credit_card (id, status) SELECT card_id, CA...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0	0.031 sec
31	11:02:21	SELECT * FROM db_status_credit_card	275 row(s) returned	0.000 sec / 0.000 sec

Muestra las 275 tarjetas del banco de datos, ahora con el estatus actualizado.

EJERCICIO 01

¿Cuántas tarjetas están activas?

Para esta consulta, si ha hecho el conteo de las tarjetas que tienen el estatus 'activated'

The screenshot shows a database interface with the following details:

- Schemas:** A tree view showing schemas: biblioteca, classicmodels, and db_sp04_transaction. The db_sp04_transaction schema is expanded, showing tables: db_company, db_credit_card, db_status_credit_card, db_transactions, and db_user.
- Query Editor:** Displays two SQL statements:

```
205 •  SELECT *  
206   FROM db_status_credit_card;  
208  
209 •  SELECT COUNT(status) AS Activated  
210   FROM db_status_credit_card  
211   WHERE status = 'activated';  
212  
213
```
- Result Grid:** Shows the result of the second query:

Activated
275
- Action Output:** Shows the log of actions taken:

#	Time	Action	Message	Duration / Fetch
31	11:02:21	SELECT * FROM db_status_credit_card	275 row(s) returned	0.000 sec / 0.000 sec
32	11:03:14	SELECT COUNT(status) AS Activated FROM db_status_credit_card WHERE...	1 row(s) returned	0.000 sec / 0.000 sec

Como resultado, tenemos que todas (275) las tarjetas están activas (activated).

NIVEL 3

Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes producto_ids. Genera la siguiente consulta:

Inicialmente, revisamos los datos de la tabla products:

TABLA PRODUCTS				
COLUMNA	DATOS A INSERIR	TIPOLOGIA	USADO	RELACIÓN
<i>id</i>	1	Entero	VARCHAR(255): mismo que en db_transactions.	PRIMARY KEY
product_name	Direwolf Stannis	caracteres alfanuméricos.	VARCHAR(255)	
price	\$161.11	Decimal, presenta los precios	DECIMAL(10, 2)	
colour	#7c7c7c	caracteres alfanuméricos.	VARCHAR(255)	
weight	1.5	Float o decimal	DECIMAL(10, 2)	
warehouse_id	WH-4	caracteres alfanuméricos.	VARCHAR(255)	

Creación de la tabla db_products:

```

-- Creación tabla products:
CREATE TABLE IF NOT EXISTS db_products (
    id VARCHAR(255) PRIMARY KEY,
    product_name VARCHAR(255),
    price DECIMAL(10,2),
    colour VARCHAR(255),
    weight DECIMAL(10,2),
    warehouse_id VARCHAR(255)
);

```

	id	product_name	price	colour	weight	warehouse_id
*	NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
41	11:10:29	CREATE TABLE IF NOT EXISTS db_products (id VARCHAR(255) PRIM...	0 row(s) affected	0.047 sec
42	11:10:33	SELECT * FROM db_products	0 row(s) returned	0.000 sec / 0.000 sec

Añadimos los datos del Excel e inicialmente nos da el error:

Error Code: 1366. Incorrect decimal value: '\$161.11' for column 'price' at row 1.

```

SCHEMAS
Filter objects
biblioteca
classicmodels
db_sp04_transaction
  Tables
    db_company
    db_credit_card
    db_products
    db_status_credit_card
    db_transactions
    db_user
  Views
  Shared Procedures
Administration Schemas
Information
Schema: db_sp04_transaction

221   price DECIMAL(10,2),
222   colour VARCHAR(255),
223   weight DECIMAL(10,2),
224   warehouse_id VARCHAR(255)
225 );
226
227 • SELECT *
228   FROM db_products;
229
230 -- Carga datos excel:
231 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
232   INTO TABLE db_products
233   FIELDS TERMINATED BY ','
234   ENCLOSED BY ""
235   IGNORE 1 ROWS;
236
237

Output
Action Output
# Time Action
36 11:07:17 SELECT * FROM db_products
0 row(s) returned
0.000 sec / 0.000 sec
37 11:08:36 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/... Error Code: 1366. Incorrect decimal value: '$161.11' for column 'price' at row 1 0.047 sec

Object Info Session
Query interrupted

```

En este caso, modificamos la tabla, para que los datos de la columna Price sean VARCHAR(255).

```

SCHEMAS
Filter objects
biblioteca
classicmodels
db_sp04_transaction
  Tables
    db_company
    db_credit_card
    db_products
    db_status_credit_card
    db_transactions
    db_user
  Views
  Shared Procedures
Administration Schemas
Information
Table: db_products

Columns:
id varchar(255) PK
product_name varchar(255)
price varchar(255)
colour varchar(255)
weight decimal(10,2)
warehouse_id varchar(255)

Object Info Session
Query Completed

```

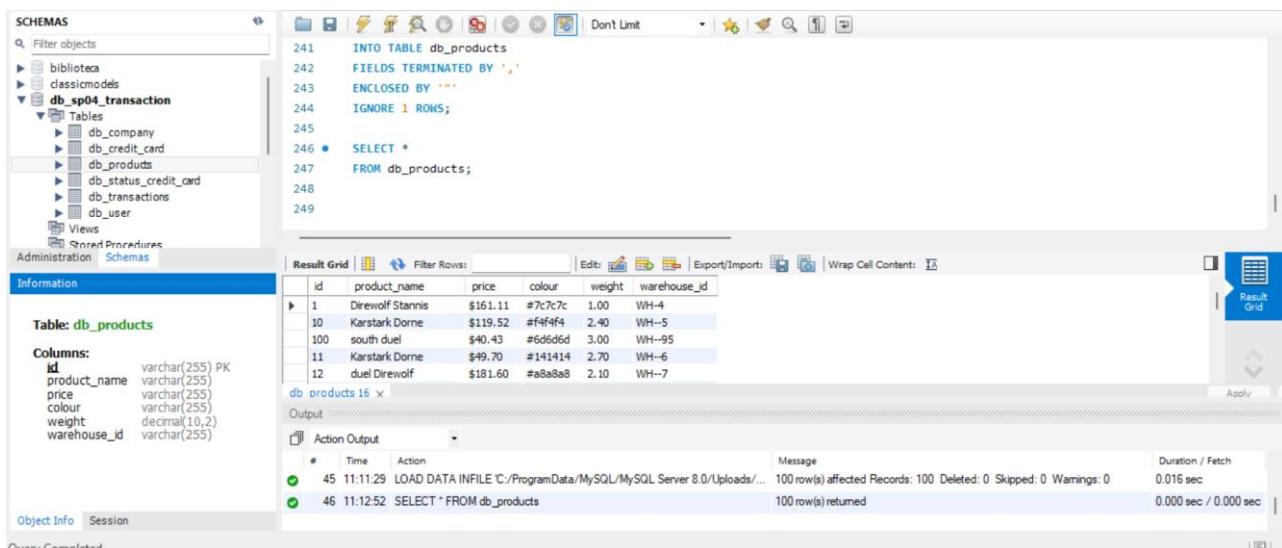
```

229
230 -- Carga datos excel:
231 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
232   INTO TABLE db_products
233   FIELDS TERMINATED BY ','
234   ENCLOSED BY ""
235   IGNORE 1 ROWS;
236
237 • ALTER TABLE db_products
238   MODIFY COLUMN price VARCHAR(255);
239
240 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
241   INTO TABLE db_products
242   FIELDS TERMINATED BY ','
243   ENCLOSED BY ""
244   IGNORE 1 ROWS;
245

Output
Action Output
# Time Action
44 11:11:24 ALTER TABLE db_products MODIFY COLUMN price VARCHAR(255)
0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0
0.078 sec
45 11:11:29 LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/... 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
0.016 sec


```

Verificamos los datos de la tabla:



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. In the center, a query editor window displays the following SQL code:

```

241 INTO TABLE db_products
242 FIELDS TERMINATED BY ','
243 ENCLOSED BY """
244 IGNORE 1 ROWS;
245
246 • SELECT *
247 FROM db_products;
248
249

```

Below the code, the 'Result Grid' shows the data from the db_products table:

	id	product_name	price	colour	weight	warehouse_id
▶	1	Direwolf Stannis	\$161.11	#7c7c7c	1.00	WH-4
10	Karstark Dorne	\$119.52	#f4f4f4	2.40	WH-5	
100	south duel	\$40.43	#6d6d6d	3.00	WH-95	
11	Karstark Dorne	\$49.70	#141414	2.70	WH-6	
12	duel Direwolf	\$181.60	#a8a8a8	2.10	WH-7	

The 'Information' panel on the left provides details about the db_products table, including its columns:

Table: db_products

Columns:

- id** varchar(255) PK
- product_name** varchar(255)
- price** varchar(255)
- colour** varchar(255)
- weight** decimal(10,2)
- warehouse_id** varchar(255)

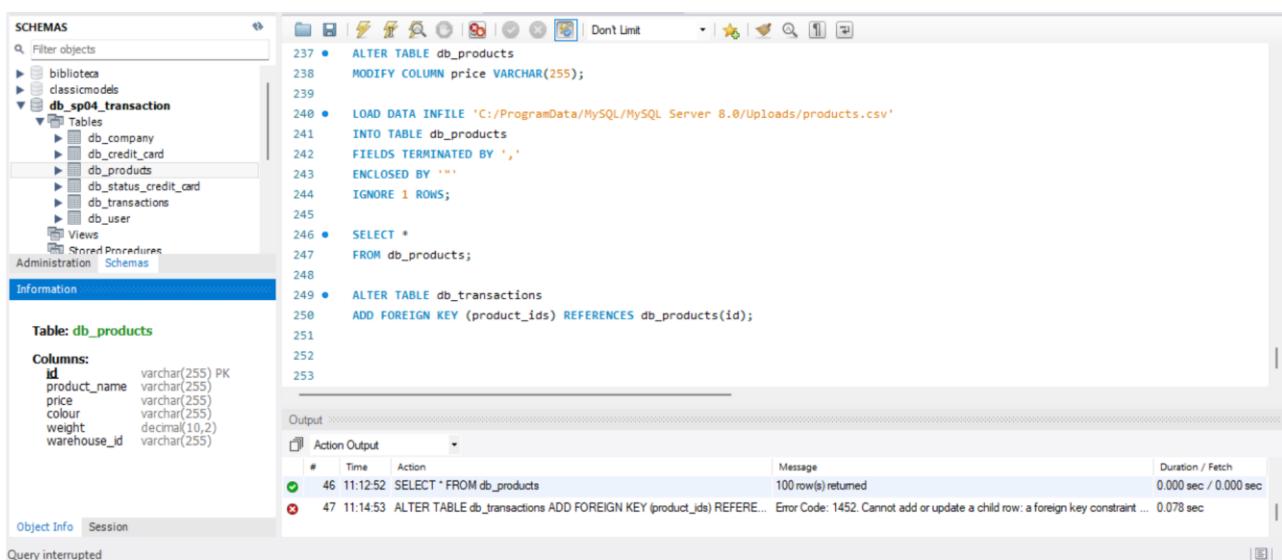
The 'Object Info' and 'Session' tabs are at the bottom.

Tenemos 100 productos registrados en el database.

Para hacer la consulta, falta crear la clave foránea en la tabla db_transactions para que las dos tablas puedan relacionarse:

En este caso nos dio en siguiente error:

*Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails
(`db_sp04_transaction`.`#sql-10cc_9`, CONSTRAINT `db_transactions_ibfk_4` FOREIGN KEY
(`product_ids`) REFERENCES `db_products`(`id`))*



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. In the center, a query editor window displays the following SQL code:

```

237 • ALTER TABLE db_products
238   MODIFY COLUMN price VARCHAR(255);
239
240 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
241   INTO TABLE db_products
242   FIELDS TERMINATED BY ','
243   ENCLOSED BY """
244   IGNORE 1 ROWS;
245
246 • SELECT *
247   FROM db_products;
248
249 • ALTER TABLE db_transactions
250   ADD FOREIGN KEY (product_ids) REFERENCES db_products(id);
251
252
253

```

The 'Information' panel on the left provides details about the db_products table, including its columns:

Table: db_products

Columns:

- id** varchar(255) PK
- product_name** varchar(255)
- price** varchar(255)
- colour** varchar(255)
- weight** decimal(10,2)
- warehouse_id** varchar(255)

The 'Object Info' and 'Session' tabs are at the bottom.

In the 'Output' panel, the last two lines show the error:

- Action Output: # 46 11:12:52 SELECT * FROM db_products Message: 100 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
- Action Output: # 47 11:14:53 ALTER TABLE db_transactions ADD FOREIGN KEY (product_ids) REFEREN... Error Code: 1452. Cannot add or update a child row: a foreign key constraint ... Duration / Fetch: 0.078 sec

Mirando los id en las dos tablas, percibimos que en db_transactions, los products_ids está en el formato: '83, 97, 23, 71'. Lo que ya habíamos visto en fase inicial de la creación del database. En este caso, no es posible crear la clave foránea, visto que existe más de un id de producto por transaction.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree, with 'db_sp04_transaction' selected. Under 'Tables', 'db_products' is selected, showing its columns: id (PK), product_name, price, colour, weight, and warehouse_id. The main area contains an SQL editor with the following code:

```
245
246 • SELECT *
247   FROM db_products;
248
249 • ALTER TABLE db_transactions
250   ADD FOREIGN KEY (product_ids) REFERENCES db_products(id);
251
252 • SELECT product_ids
253   FROM db_transactions;
```

The 'Result Grid' tab is active, showing the output of the last query:

product_ids
71, 1, 19
47, 97, 43
47, 67, 31, 5
89, 83, 79
43, 31

The 'Session' tab at the bottom shows the following history:

#	Time	Action	Message	Duration / Fetch
47	11:14:53	ALTER TABLE db_transactions ADD FOREIGN KEY (product_ids) REFERERE...	Error Code: 1452. Cannot add or update a child row: a foreign key constraint ...	0.078 sec
48	11:16:06	SELECT product_ids FROM db_transactions	587 row(s) returned	0.000 sec / 0.000 sec

Para solucionarlo, se crea una nueva tabla (puente), con los datos de los productos y su respectiva transacción, en formato único, para eliminar los ids múltiples en product_ids, de la tabla db_transactions.

Creación de la tabla db_transaction_product:

The screenshot shows the MySQL Workbench interface. The left sidebar displays the schema structure under 'Schemas'. In the 'Tables' section of the 'db_sp04_transaction' schema, the 'db_transaction_product' table is selected. The main pane shows the SQL code for creating the table:

```
255 • CREATE TABLE db_transaction_product (
256     transaction_id VARCHAR(255),
257     product_id VARCHAR(255),
258     PRIMARY KEY (transaction_id, product_id),
259     FOREIGN KEY (transaction_id) REFERENCES db_transactions(id),
260     FOREIGN KEY (product_id) REFERENCES db_products(id)
261 );
262
263 • SELECT *
264   FROM db_transaction_product;
```

Below the SQL editor is the 'Result Grid' showing the table's structure:

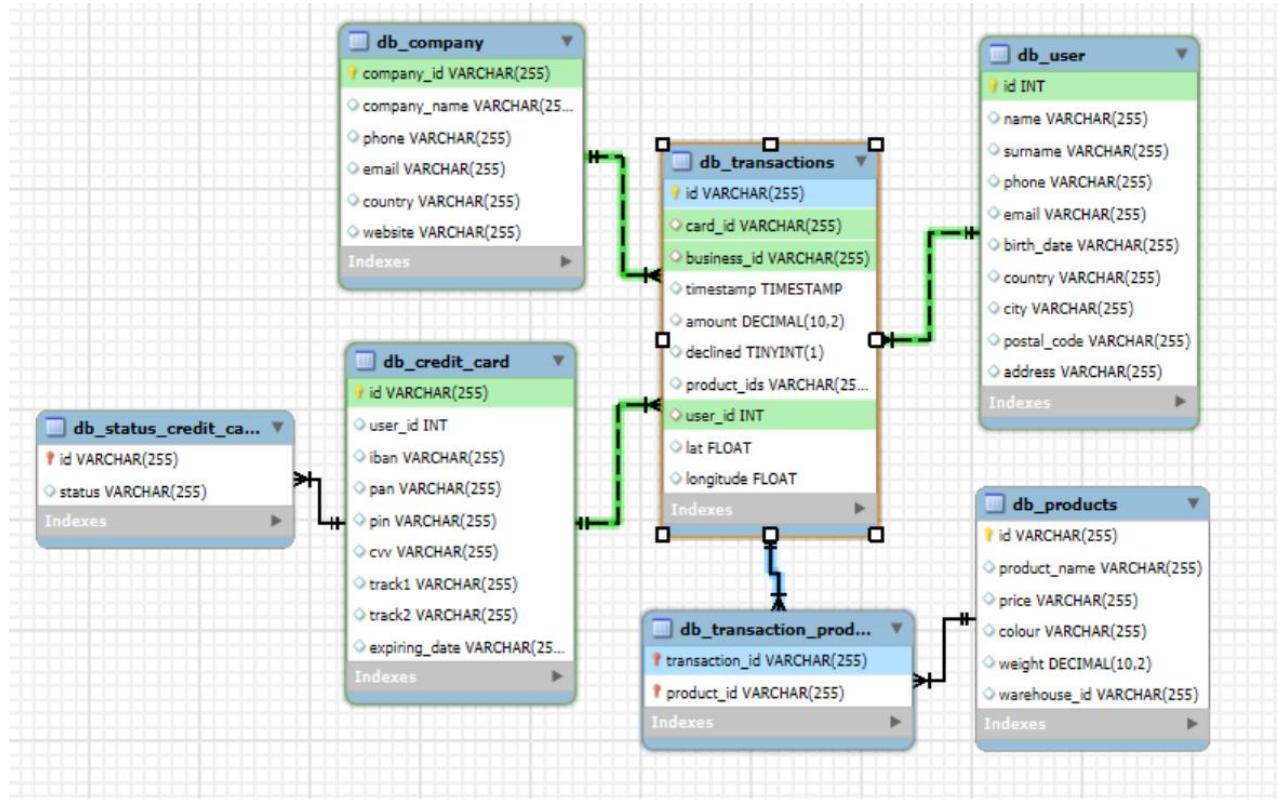
transaction_id	product_id
NULL	NULL

At the bottom, the 'Action Output' pane shows the results of the last two queries:

#	Time	Action	Message	Duration / Fetch
54	11:26:18	CREATE TABLE db_transaction_product (transaction_id VARCHAR(255)...)	0 row(s) affected	0.078 sec
55	11:26:26	SELECT * FROM db_transaction_product	0 row(s) returned	0.000 sec / 0.000 sec

El modelo relacional sigue con la tabla de hechos es db_transactions (1:N), con las dimensiones users, credit_card y companies. Luego, se ha creado dos tablas puentes, para verificar el estatus de las tarjetas (db_status_credit_card) y una tabla puente (db_transaction_product) para verificar los

ids de los productos, relacionada directamente entre db_transactions y db_products. Este es el modelo finalizado:



Verificamos las claves foráneas del modelo:

The screenshot shows the MySQL Workbench Schema Editor interface. A new schema named 'db_sp04_transaction' is being created. The schema details are as follows:

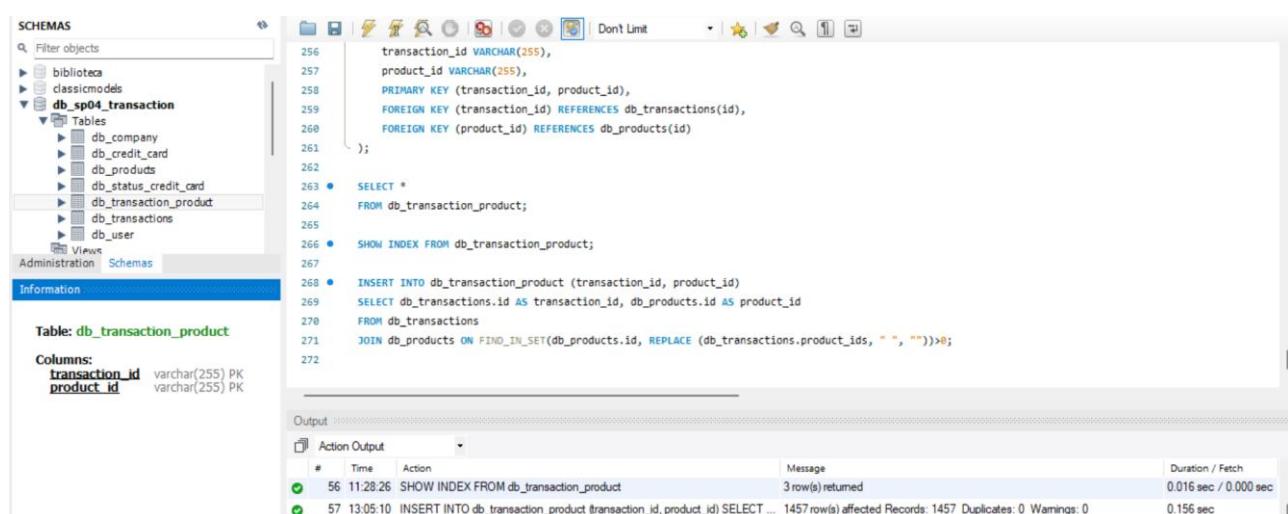
- Name:** db_sp04_transaction
- Charset/Collation:** utf8mb4 / utf8mb4_0900_
- Comments:** (empty)

The 'Layers' tab in the left sidebar is selected, showing the layers: db_transaction_product, db_transactions, db_user, and db_products. The 'Schema' tab is active at the bottom.

Por último, para insertar los datos y organizar los ids de productos de db_transactions para que sean únicos, se ha visto que existen muchas posibilidades. Inicialmente, se ha probado el STRING_SPLIT, pero solo funciona en SQL Server y no en MySQL. (fuente: [Microsoft](#), [youtube](#)). En la peer revisión, si ha visto las funciones WITH RECURSIVE, TRIM y SUBSTRING INDEX ([learnsql](#), [galileu](#), [MySQLManual](#), [MySQLManual-substring](#)).

Por fin, se ha decidido utilizar la función FIND_IN_SET y REPLACE ([MySQLTutorial](#), [MySQLManual](#)), que pareció más sencilla, de esta forma se ha creado este código:

- **INSERT TO db_transaction_product (transaction_id, product_id):** comando para insertar el resultado en las columnas (transaction_id, product_id) de la nueva tabla (db_transaction_product).
- **SELECT db_transactions.id AS transaction_id, db_products.id AS product_id:** indicamos las columnas de las tablas db_transactions y db_products, donde queremos que se inserten los datos a la nueva tabla (db_transaction_product).
- **FROM db_transactions:** especificamos cual es la tabla principal para sacar los datos.
- **JOIN db_products:** unimos las dos tablas, con la condición a seguir
- **ON FIND_IN_SET(db_products.id, REPLACE(db_transactions.product_ids, " ", ""))>0:** db_transactions.product_ids es la columna que tiene multiplos ids, separados por coma. El " ", "" quita los espacios en blanco dentro de la string. La función FIND_IN_SET verifica si el id del producto está presente en la lista product_ids de la transacción. La función devuelve un valor mayor que 0 si el producto es encontrado, o 0 si no es encontrado. El >0 asegura que solo se seleccionen las filas donde el producto existe en la lista de product_ids.



The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'db_sp04_transaction' schema selected. Under 'Tables', the 'db_transaction_product' table is highlighted. Below the table list, the 'Information' tab is active, showing the table definition and its columns: 'transaction_id' and 'product_id'. The 'transaction_id' column is defined as VARCHAR(255) and has a PRIMARY KEY constraint. The 'product_id' column is also defined as VARCHAR(255). The 'Columns' section shows the same two columns. The 'Output' tab at the bottom displays the SQL code used to create the table and the results of the 'SHOW INDEX' and 'INSERT' statements. The 'Action Output' section shows the execution details, including the time, action, message, duration, and fetch count.

```

CREATE TABLE `db_transaction_product` (
  `transaction_id` varchar(255) NOT NULL,
  `product_id` varchar(255) NOT NULL,
  PRIMARY KEY (`transaction_id`, `product_id`),
  FOREIGN KEY (`transaction_id`) REFERENCES `db_transactions`(`id`),
  FOREIGN KEY (`product_id`) REFERENCES `db_products`(`id`)
);

SELECT *
FROM db_transaction_product;

SHOW INDEX FROM db_transaction_product;

INSERT INTO db_transaction_product (transaction_id, product_id)
SELECT db_transactions.id AS transaction_id, db_products.id AS product_id
FROM db_transactions
JOIN db_products ON FIND_IN_SET(db_products.id, REPLACE(db_transactions.product_ids, " ", ""))>0;
  
```

#	Time	Action	Message	Duration / Fetch
56	11:28:26	SHOW INDEX FROM db_transaction_product	3 row(s) returned	0.016 sec / 0.000 sec
57	13:05:10	INSERT INTO db_transaction_product (transaction_id, product_id) SELECT ... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0		0.156 sec

Ahora verificamos que los ids, hayan sido ajustados de forma correcta:

```

265 • SHOW INDEX FROM db_transaction_product;
266 • INSERT INTO db_transaction_product (transaction_id, product_id)
267     SELECT db_transactions.id AS transaction_id, db_products.id AS product_id
268     FROM db_transactions
269     JOIN db_products ON FIND_IN_SET(db_products.id, REPLACE (db_transactions.product_ids, " ", ""))>0;
270
271 • SELECT *
272     FROM db_transaction_product;
273
274

```

Result Grid:

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
0466A42E-47CF-8D24-FD01-C08689713128	43
0466A42E-47CF-8D24-FD01-C08689713128	47

Action Output:

#	Time	Action	Message	Duration / Fetch
59	13:07:25	INSERT INTO db_transaction_product (transaction_id, product_id) SELECT ...	Error Code: 1062: Duplicate entry '02C6201E-D90A-1859-B4EE-88D2986D3...' for key 'transaction_id'.	0.000 sec
60	13:09:48	SELECT * FROM db_transaction_product	1457 row(s) returned	0.000 sec / 0.000 sec

Verificamos la transacción: '02C6201E-D90A-1859-B4EE-88D2986D3B02':

```

269 • SELECT db_transactions.id AS transaction_id, db_products.id AS product_id
270     FROM db_transactions
271     JOIN db_products ON FIND_IN_SET(db_products.id, REPLACE (db_transactions.product_ids, " ", ""))>0;
272
273 • SELECT *
274     FROM db_transaction_product;
275
276 • SELECT product_ids
277     FROM db_transactions
278     WHERE id = '02C6201E-D90A-1859-B4EE-88D2986D3B02';

```

Result Grid:

product_ids
71, 1, 19

Action Output:

#	Time	Action	Message	Duration / Fetch
62	13:13:23	SELECT product_ids FROM db_transactions WHERE transaction_id = '02C...'	Error Code: 1054: Unknown column 'transaction_id' in 'where clause'	0.000 sec
63	13:13:35	SELECT product_ids FROM db_transactions WHERE id = '02C6201E-D90A...'	1 row(s) returned	0.000 sec / 0.000 sec

Aparecen los ids 1, 19 y 71, como en la imagen anterior, se entiende que la función FIND_IN_SET dividió los ids de la tabla db_transactions y luego el REPLACE reemplazó el id en una nueva línea con el mismo transaction_id, pero con solo un id de producto. En este caso específico eran 3 productos diferentes y creó 3 líneas.

EJERCICIO 01

Necesitamos conocer el número de veces que se ha vendido cada producto.

Para esta consulta, con la nueva tabla db_transaction_product, hacemos el COUNT de transactions para encontrar cuantas veces se ha vendido cada producto.

```

279
280    -- Necesitamos conocer el número de veces que se ha vendido cada producto.
281 •     SELECT pr.id, pr.product_name, COUNT(tr.transaction_id) AS Num_Ventas
282     FROM db_transaction_product tr
283     JOIN db_products pr
284     ON tr.product_id = pr.id
285     GROUP BY 1, 2
286     ORDER BY 3 DESC;
287
288

```

	id	product_name	Num_Ventas
▶	23	riverrlands north	68
▶	67	Winterfell	68
▶	79	Direwolf riverrlands the	66
▶	2	Tarly Stark	65
▶	43	duel	65

Result 27 x

Action Output

#	Time	Action	Message	Duration / Fetch
65	13:19:18	SELECT pr.id, pr.product_name, COUNT(tr.transaction_id) AS Num_Ventas ...	26 row(s) returned	0.015 sec / 0.000 sec
66	13:19:26	SELECT pr.id, pr.product_name, COUNT(tr.transaction_id) AS Num_Ventas ...	26 row(s) returned	0.015 sec / 0.000 sec

Retornan 26 líneas, con el conteo de ventas de cada producto. Se percibe, que hay productos que no tienen ninguna transacción.