

## **SPRINT 4: CREACIÓN DE BASES DE DATOS:**

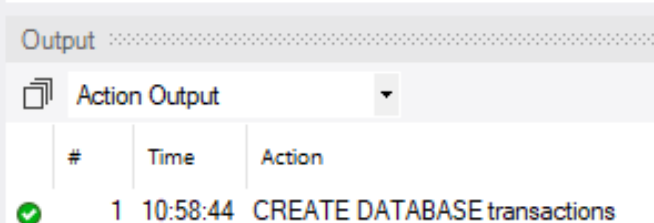
### **Nivel 1:**

Partiendo de algunos archivos CSV diseñarás y crearás tu base de datos.

Descarga los archivos CSV, estúdialas y diseña una base de datos con un esquema de estrella que contenga al menos 4 tablas.

Para iniciar se crea una base de datos vacía, para posteriormente crear y añadir las tablas, para ello se emplea una query muy sencilla como se muestra a continuación:

```
1      #creacion de la base de datos:
2      • CREATE DATABASE transactions;
3
4
```



The screenshot shows a SQL editor with the query to create a database. Below the editor, the 'Output' pane displays the 'Action Output' table, which contains one row indicating the successful creation of the 'transactions' database at 10:58:44.

#	Time	Action
1	10:58:44	CREATE DATABASE transactions

Una vez creada la base de datos transactions, como se muestra anteriormente; se continúa con la creación de las 6 tablas. A continuación, se muestra una serie de capturas de pantalla en las que se muestran las queries empleadas para la creación de cada una de las tablas. Resaltar que, en este paso, se crean las tablas vacías para posteriormente cargar los datos proporcionados en los csv.

- Creación de la tabla american\_users:

```
5      #crear tablas:
6      #american_users:
7      • CREATE TABLE IF NOT EXISTS american_users (
8          id INT PRIMARY KEY,
9          name VARCHAR(100),
10         surname VARCHAR(100),
11         phone VARCHAR(150),
12         email VARCHAR(150),
13         birth_date VARCHAR(100),
14         country VARCHAR(150),
15         city VARCHAR(150),
16         postal_code VARCHAR(100),
17         address VARCHAR(255)
18     );
```



The screenshot shows a SQL editor with the query to create a table. Below the editor, the 'Output' pane displays the 'Action Output' table, which contains two rows indicating the successful execution of two SQL actions at 11:08:54 and 11:09:09.

#	Time	Action	Message
1	11:08:54	CREATE DATABASE transactions	1 row(s) affected
2	11:09:09	CREATE TABLE IF NOT EXISTS american_users ( id INT PRIMARY KEY, name VA...	0 row(s) affected

- Creación de la tabla european\_users:

```

19  #european_users:
20  ● ○ CREATE TABLE IF NOT EXISTS european_users (
21      id INT PRIMARY KEY,
22      name VARCHAR(100),
23      surname VARCHAR(100),
24      phone VARCHAR(150),
25      email VARCHAR(150),
26      birth_date VARCHAR(100),
27      country VARCHAR(150),
28      city VARCHAR(150),
29      postal_code VARCHAR(100),
30      address VARCHAR(255)
31  );
32
33

```

Output

Action Output

#	Time	Action	Message
✓ 1	11:08:54	CREATE DATABASE transactions	1 row(s) affected
✓ 2	11:09:09	CREATE TABLE IF NOT EXISTS american_users ( id INT PRIMARY KEY, name VA...	0 row(s) affected
✓ 3	11:10:54	CREATE TABLE IF NOT EXISTS european_users ( id INT PRIMARY KEY, name VA...	0 row(s) affected

Para estas dos tablas, es importante resaltar que cuentan con los mismos campos.

- Creación de la tabla companies:

```

34  #companies:
35  ● ○ CREATE TABLE IF NOT EXISTS companies (
36      company_id VARCHAR(15) PRIMARY KEY,
37      company_name VARCHAR(100),
38      phone VARCHAR(15),
39      email VARCHAR(100),
40      country VARCHAR(100),
41      website VARCHAR(255)
42  );
43

```

Output

Action Output

#	Time	Action	Message
✓ 1	11:08:54	CREATE DATABASE transactions	1 row(s) affected
✓ 2	11:09:09	CREATE TABLE IF NOT EXISTS american_users ( id INT PRIMARY KEY, name VA...	0 row(s) affected
✓ 3	11:10:54	CREATE TABLE IF NOT EXISTS european_users ( id INT PRIMARY KEY, name VA...	0 row(s) affected
✓ 4	11:11:58	CREATE TABLE IF NOT EXISTS companies ( company_id VARCHAR(15) PRIMARY KEY...	0 row(s) affected

- Creación de la tabla credit\_cards:

```

--
45     #credit_cards
46 ● ○ CREATE TABLE IF NOT EXISTS credit_cards (
47     id VARCHAR(20) PRIMARY KEY,
48     user_id INT,
49     iban VARCHAR(50),
50     pan VARCHAR(25),
51     pin VARCHAR(4),
52     cvv VARCHAR(15), #no es int para que no elimine los que inician por 0
53     track1 VARCHAR (255),
54     track2 VARCHAR (255),
55     expiring_date VARCHAR (255)
56 );
--

```

- Creación de la tabla products:

```

57     #products
58 ● ○ CREATE TABLE IF NOT EXISTS products (
59     id INT PRIMARY KEY,
60     product_name VARCHAR(50),
61     price DECIMAL(10, 2),
62     colour VARCHAR(50),
63     weight DECIMAL(5, 2),
64     warehouse_id VARCHAR(15)
65 );
66
67

```

Output				
Action Output				
#	Time	Action		Message
✓ 1	11:08:54	CREATE DATABASE transactions		1 row(s) affected
✓ 2	11:09:09	CREATE TABLE IF NOT EXISTS american_users (	id INT PRIMARY KEY, name VA...	0 row(s) affected
✓ 3	11:10:54	CREATE TABLE IF NOT EXISTS european_users (	id INT PRIMARY KEY, name VA...	0 row(s) affected
✓ 4	11:11:58	CREATE TABLE IF NOT EXISTS companies (	company_id VARCHAR(15) PRIMARY KEY...	0 row(s) affected
✓ 5	11:12:52	CREATE TABLE IF NOT EXISTS credit_cards (	id VARCHAR(20) PRIMARY KEY, user_id IN...	0 row(s) affected
✓ 6	11:13:35	CREATE TABLE IF NOT EXISTS products (	id INT PRIMARY KEY, product_name V...	0 row(s) affected

- Creación de la tabla transactions:

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' tab is active, showing a tree view of databases: sakila, sys, tienda\_online, and transactions. The 'transactions' database is selected, and its 'Tables' folder is expanded, showing tables like american\_users, companies, credit\_cards, european\_users, products, and transactions. The 'Information' tab is also visible, showing the 'Schema: transactions'.

The main editor shows a SQL script to create the 'transactions' table:

```

68 #transactions
69 CREATE TABLE IF NOT EXISTS transactions (
70     id VARCHAR(255) PRIMARY KEY,
71     card_id VARCHAR(20),
72     business_id VARCHAR(20),
73     timestamp TIMESTAMP,
74     amount DECIMAL(10, 2),
75     declined TINYINT,
76     product_ids VARCHAR(50),
77     user_id INT,
78     lat FLOAT,
79     longitude FLOAT
80 );

```

The 'Output' tab is active, showing the 'Action Output' table with the following data:

#	Time	Action	Message
2	11:09:09	CREATE TABLE IF NOT EXISTS american_users ( id INT PRIMARY KEY, name V...	0 row(s) affected
3	11:10:54	CREATE TABLE IF NOT EXISTS european_users ( id INT PRIMARY KEY, name V...	0 row(s) affected
4	11:11:58	CREATE TABLE IF NOT EXISTS companies ( company_id VARCHAR(15) PRIMARY K...	0 row(s) affected
5	11:12:52	CREATE TABLE IF NOT EXISTS credit_cards (id VARCHAR(20) PRIMARY KEY, user_id l...	0 row(s) affected
6	11:13:35	CREATE TABLE IF NOT EXISTS products ( id INT PRIMARY KEY, product_name ...	0 row(s) affected
7	11:14:19	CREATE TABLE IF NOT EXISTS transactions ( id VARCHAR(255) PRIMARY KEY, ...	0 row(s) affected

Se continuó con la carga de los datos a cada tabla. Después de muchos intentos, muchos errores, pruebas de ensayo y error, se logró cargar los datos; aparentemente había que tener cierta accesibilidad para cargar datos desde una carpeta en concreto. Para localizar dicha carpeta y activar los permisos necesarios, se emplearon las líneas de código que se muestran a continuación en la captura de pantalla junto con los muchos outputs de ensayo y error hasta que se consiguió.

The screenshot shows the MySQL Workbench interface. The main editor shows a SQL script to set file permissions:

```

83 #Cargar los datos en las tablas:
84 show variables like "secure_file_priv";
85 -- 'secure_file_priv', 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\'
86 show global variables like "local_infile";
87 SET GLOBAL local_infile = 1;
88 SHOW GRANTS FOR CURRENT_USER();
89 GRANT FILE ON *.* TO 'root'@'localhost';
90 FLUSH PRIVILEGES;

```

The 'Output' tab is active, showing the 'Action Output' table with the following data:

#	Time	Action	Message
1	13:11:28	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on .
2	13:12:50	show global variables like "local_infile"	1 row(s) returned
3	13:12:55	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on .
4	13:13:01	GRANT FILE ON *.* TO 'root'@'localhost'	0 row(s) affected
5	13:13:04	LOAD DATA LOCAL INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on .
6	13:13:24	SET GLOBAL local_infile = 1	0 row(s) affected

83	#Cargar los datos en las tablas:
84	• <code>show variables like "secure_file_priv";</code>
85	-- 'secure_file_priv', 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\'
86	• <code>show global variables like "local_infile";</code>
87	• <code>SET GLOBAL local_infile = 1;</code>
88	• <code>SHOW GRANTS FOR CURRENT_USER();</code>
89	• <code>GRANT FILE ON *.* TO 'root'@'localhost';</code>

Output				
Action Output				
#	Time	Action	Message	
7	13:13:26	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on ...	
8	13:13:30	FLUSH PRIVILEGES	0 row(s) affected	
9	13:13:33	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on ...	
10	13:14:12	show variables like "secure_file_priv"	1 row(s) returned	
11	13:14:20	show global variables like "local_infile"	1 row(s) returned	
12	13:14:23	SET GLOBAL local_infile = 1	0 row(s) affected	
13	13:14:27	SHOW GRANTS FOR CURRENT_USER()	3 row(s) returned	
14	13:14:36	GRANT FILE ON *.* TO 'root'@'localhost'	0 row(s) affected	
15	13:14:38	FLUSH PRIVILEGES	0 row(s) affected	
16	13:14:44	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on ...	
17	13:18:13	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\american_us...	Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cann...	
18	13:19:51	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_us...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0	
19	13:20:21	SELECT * FROM transactions.american_users	1010 row(s) returned	
20	13:23:19	show global variables like "local_infile"	1 row(s) returned	

Una vez logrados los permisos, haber localizado la carpeta y los “/” para la accesibilidad correcta, se prosiguió a cargar los datos proporcionados en los csv a cada una de las tablas correspondientes, anteriormente creadas. Las queries que se emplearon para la carga de los datos en cada tabla se muestran a continuación:

- Carga de datos para la tabla american\_users:

92	#tabla american_users:
93	• <code>LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_users.csv'</code>
94	<code>INTO TABLE american_users</code>
95	<code>FIELDS TERMINATED BY ','</code>
96	<code>ENCLOSED BY ''</code>
97	<code>LINES TERMINATED BY '\n'</code>
98	<code>IGNORE 1 ROWS</code>
99	<code>(id,name,surname,phone,email,birth_date,country,city,postal_code,address);</code>
100	

Output				
Action Output				
#	Time	Action	Message	
10	13:14:12	show variables like "secure_file_priv"	1 row(s) returned	
11	13:14:20	show global variables like "local_infile"	1 row(s) returned	
12	13:14:23	SET GLOBAL local_infile = 1	0 row(s) affected	
13	13:14:27	SHOW GRANTS FOR CURRENT_USER()	3 row(s) returned	
14	13:14:36	GRANT FILE ON *.* TO 'root'@'localhost'	0 row(s) affected	
15	13:14:38	FLUSH PRIVILEGES	0 row(s) affected	
16	13:14:44	LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\ameri...	Error Code: 2068. LOAD DATA LOCAL INFILE file request rejected due to restrictions on ...	
17	13:18:13	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\american_us...	Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cann...	
18	13:19:51	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_us...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0	
19	13:20:21	SELECT * FROM transactions.american_users	1010 row(s) returned	
20	13:23:19	show global variables like "local_infile"	1 row(s) returned	
21	13:53:56	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_us...	3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0	

1 • `SELECT * FROM transactions.american_users;`

**Result Grid** | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch rows:

	id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble		1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	New York	10001	348-7818 Sagittis St.
2	Garrett	Mcconnell		(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Philadelphia	19101	903 Sit Ave
3	Claran	Harrison		(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Houston	77001	736-2063 Tellus St.
4	Howard	Stafford		1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Phoenix	85001	Ap #545-2244 Erat. Rd.
5	Hayfa	Pierce		1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Philadelphia	19101	341-2821 Ultrices Av.
6	Joel	Tyson		(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	San Jose	95101	888-2799 Amet Street
7	Rafael	Jimenez		(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Chicago	60601	8627 Malesuada Rd.
8	Nissim	Franks		(692) 157-3469	egestas.aliquam.fringilla@google.ca	Aug 1, 1993	United States	New York	10001	Ap #251-7144 Integer St.
9	Mannix	Mcclain		(590) 883-2184	aliquam.nisl@outlook.com	Jan 24, 1987	United States	San Antonio	78201	647-3080 Lacus. St.
10	Robert	Mccarthy		(324) 746-6771	fermentum@protonmail.com	Apr 30, 1984	United States	San Jose	95101	P.O. Box 773
11	Joan	Baird		(981) 429-8106	et@outlook.net	Feb 25, 1990	United States	Los Angeles	90001	P.O. Box 687
12	Benedict	Wheeler		1-515-824-2855	tincidunt.donec.vitae@hotmail.couk	Aug 6, 1999	United States	Phoenix	85001	748-8694 Porttitor Avenue
13	Allegra	Stanton		1-927-753-6488	proin.eget@protonmail.ca	May 19, 1990	United States	New York	10001	4457 Ante. Av.
14	Sara	Flynn		1-311-646-9333	integer@outlook.net	Dec 27, 1988	United States	Los Angeles	90001	P.O. Box 865

Output

Action Output

#	Time	Action	Message
✓ 19	13:20:21	SELECT * FROM transactions.american_users	1010 row(s) returned
✓ 20	13:23:19	show global variables like "local_infile"	1 row(s) returned

- Carga de datos para la tabla european\_users:

```

101 #tabla european_users:
102 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_users.csv'
103 INTO TABLE european_users
104 FIELDS TERMINATED BY ','
105 ENCLOSED BY '"'
106 LINES TERMINATED BY '\n'
107 IGNORE 1 ROWS
108 (id,name,surname,phone,email,birth_date,country,city,postal_code,address);
109
110

```

Output

Action Output

#	Time	Action	Message
✗ 17	13:18:13	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_us...	Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cann...
✓ 18	13:19:51	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_us...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0
✓ 19	13:20:21	SELECT * FROM transactions.american_users	1010 row(s) returned
✓ 20	13:23:19	show global variables like "local_infile"	1 row(s) returned
✓ 21	13:53:56	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_us...	3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0

1 • `SELECT * FROM transactions.european_users;`

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

	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	London	EC1A 1BB	Ap #432-4493 Aliquet Rd.
	152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google.edu	Sep 30, 1979	United Kingdom	Birmingham	B1 1AA	551-8930 Lobortis Street
	153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	London	EC1A 1BB	Ap #312-5898 Consectetuer St.
	154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Manchester	M1 1AE	872-1866 Pede Rd.
	155	Joshua	Russell	055 4409 5286	justo.nec.ante@outlook.edu	Jan 23, 1984	United Kingdom	Manchester	M1 1AE	Ap #285-4727 Auctor. Av.
	156	Remedios	Case	055 3114 1566	mollis.phasellus.libero@aol.com	Oct 9, 1994	United Kingdom	Liverpool	L1 1AA	479-3690 Turpis Road
	157	Philip	Carey	0800 640 6251	phasellus@yahoo.net	Oct 10, 1992	United Kingdom	Manchester	M1 1AE	196-1103 Quisque Street
	158	Fatima	Dyer	0800 1111	adipiscing@google.org	Dec 24, 1988	United Kingdom	Birmingham	B1 1AA	Ap #435-7194 Scelerisque, St.
	159	Kylynn	Acevedo	056 5727 9602	dignissim.lacus.aliquam@google.org	May 10, 2000	United Kingdom	Liverpool	L1 1AA	871-3506 Lectus. Ave

Output

Action Output

#	Time	Action	Message
✓ 18	13:19:51	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_us...	1010 row(s) affected Records: 1010 Deleted: 0 Skipped: 0 Warnings: 0
✓ 19	13:20:21	SELECT * FROM transactions.american_users	1010 row(s) returned
✓ 20	13:23:19	show global variables like "local_infile"	1 row(s) returned
✓ 21	13:53:56	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_us...	3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0
✓ 22	13:57:46	SELECT * FROM transactions.european_users	3990 row(s) returned

- Cargar los datos para la tabla companies:

```

119 #tabla companies:
120 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'
121 INTO TABLE companies
122 FIELDS TERMINATED BY ','
123 ENCLOSED BY '"'
124 LINES TERMINATED BY '\n'
125 IGNORE 1 ROWS
126 (company_id,company_name,phone,email,country,website);
127

```

Output

Action Output

#	Time	Action	Message
✓ 23	14:21:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.c...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
✓ 24	14:21:11	SELECT * FROM transactions.companies	100 row(s) returned

```
1 • SELECT * FROM transactions.companies;
```

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.couk	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.couk	Norway	https://nytimes.com/user/110
b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cnn.com/one

companies 1 x

Output

Action Output

#	Time	Action	Message
23	14:21:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.c...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
24	14:21:11	SELECT * FROM transactions.companies	100 row(s) returned

- Carga de los datos en la tabla credit\_cards:

```
128 #credit_cards
129 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
130 INTO TABLE credit_cards
131 FIELDS TERMINATED BY ','
132 ENCLOSED BY '"'
133 LINES TERMINATED BY '\n'
134 IGNORE 1 ROWS
135 (id,user_id,iban,pan,pin,cvv,track1,track2,expiring_date);
136
```

Output

Action Output

#	Time	Action	Message
28	14:27:53	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards....	5000 row(s) affected Records: 5000 Deleted: 0 Skipped: 0 Warnings: 0
29	14:28:00	SELECT * FROM transactions.credit_cards	5000 row(s) returned



1 • `SELECT * FROM transactions.credit_cards;`

	id	user_id	iban	pan	pin	cvv	track1	track2	expiring_da
▶	CcS-4857	276	XX4857591835292505850771	2314242385113924	1819	467	%B2314242385113924^LWCBUDLWCBUD^22...	%B2314242385113924=2410101518363164?	09/27/25
	CcS-4858	277	XX8581768137002436094025	6582720299715533	3964	817	%B6582720299715533^TIQMVITIQMI^2404...	%B6582720299715533=2411101104546272?	12/28/28
	CcS-4859	278	XX7826930491423553609370	8861684536289642	4983	277	%B8861684536289642^COFBGDCOFBGD^280...	%B8861684536289642=2502101761665371?	11/26/26
	CcS-4860	279	XX5559590368835304645299	2481155515498459	6876	661	%B2481155515498459^TIUJTUTIUJTU^31040...	%B2481155515498459=2602101514414395?	07/27/27
	CcS-4861	280	XX2035182877195191627307	1308930301149557	5710	398	%B1308930301149557^HPOBNZHPOBNZ^330...	%B1308930301149557=2805101751305028?	04/25/26
	CcS-4862	281	XX4774721462463645409758	6715617009807829	4042	174	%B6715617009807829^LDMWTDLDMWTD^33...	%B6715617009807829=2210101702370428?	11/27/26

credit\_cards 1 x Apply

Output

Action Output

#	Time	Action	Message	Duration / Fe
28	14:27:53	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards....'	5000 row(s) affected Records: 5000 Deleted: 0 Skipped: 0 Warnings: 0	1.235 sec
29	14:28:00	SELECT * FROM transactions.credit_cards	5000 row(s) returned	0.000 sec /

- Carga de los datos en la tabla products:

Para esta tabla se encontró un error para cargar los datos

```

137 #products
138 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
139 INTO TABLE products
140 FIELDS TERMINATED BY ','
141 ENCLOSED BY '"'
142 LINES TERMINATED BY '\n'
143 IGNORE 1 ROWS
144 (id,product_name,price,colour,weight,warehouse_id);
145

```

Output

Action Output

#	Time	Action	Message
31	14:30:42	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'...	Error Code: 1366. Incorrect decimal value: '\$161.11' for column 'price' at row 1

Este error 1366 se refiere a que no coinciden los tipos de datos de la tabla creada con los del csv. Al revisar el csv es notorio que los registros de Price se presentan con el símbolo \$ y por eso no coinciden con el tipo de datos DECIMAL que se indicó al crear la tabla. Para solventar el error y hacer que coincidan los tipos de datos se empleó la siguiente query:

```

137 #products
138 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
139 INTO TABLE products
140 FIELDS TERMINATED BY ','
141 ENCLOSED BY '"'
142 LINES TERMINATED BY '\n'
143 IGNORE 1 ROWS
144 (id,product_name,@price,colour,weight,warehouse_id)
145 SET price = CAST(REPLACE(@price,'$', '') AS DECIMAL(10,2));
146 ;

```

Output

Action Output

#	Time	Action	Message
33	14:46:43	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
34	14:46:50	SELECT * FROM transactions.products	100 row(s) returned

1 • `SELECT * FROM transactions.products;`

Result Grid						
Filter Rows: <input type="text"/>						
Edit:    Export/Import:   Wrap Cell Content:						
	id	product_name	price	colour	weight	warehouse_id
▶	1	Direwolf Stannis	161.11	#7c7c7c	1.00	WH-4
	2	Tarly Stark	9.24	#919191	2.00	WH-3
	3	duel tourney Lannister	171.13	#d8d8d8	1.50	WH-2
	4	warden south duel	71.89	#111111	3.00	WH-1
	5	skywalker ewok	171.22	#bdbdbd	3.20	WH-0
	6	dooku solo	136.60	#c4c4c4	0.80	WH--1
	7	north of Casterly	63.33	#b7b7b7	0.60	WH--2
	8	Winterfell	32.37	#383838	1.40	WH--3
	9	Winterfell	76.40	#b5b5b5	1.20	WH--4
	10	Karstark Dorne	119.52	#f4f4f4	2.40	WH--5
	11	Karstark Dorne	49.70	#141414	2.70	WH--6
	12	duel Direwolf	181.60	#a8a8a8	2.10	WH--7
	13	palpatine chewbacca	139.59	#2b2b2b	1.00	WH--8

products 1 x

Output

Action Output

#	Time	Action	Message
✓ 33	14:46:43	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
✓ 34	14:46:50	SELECT * FROM transactions.products	100 row(s) returned

- Cargar los datos para la tabla transactions:

```
149 #transactions
150 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
151 INTO TABLE transactions
152 FIELDS TERMINATED BY ';'
153 ENCLOSED BY ''
154 LINES TERMINATED BY '\n'
155 IGNORE 1 ROWS
156 (id,card_id,business_id,timestamp,amount,declined,product_ids,user_id,lat,longitude);
157
```

Output			
Action Output			
#	Time	Action	Message
✓ 35	14:50:31	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions....	100000 row(s) affected Records: 100000 Deleted: 0 Skipped: 0 Warnings: 0
✓ 36	14:50:43	SELECT * FROM transactions.transactions	100000 row(s) returned

```
1 • SELECT * FROM transactions.transactions;
```

Result Grid										
Filter Rows: <input type="text"/>										
Edit:     Export/Import:   Wrap Cell Content: <input type="checkbox"/> Fetch rows:										
id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude	
00043A49-2949-494B-A5DD-A5BAE3BB19DD	CcS-9294	b-2458	2024-08-28 07:16:46	395.43	0	16, 26, 97, 87	4713	46.1999	1.43554	
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	CcS-5019	b-2370	2016-12-21 20:07:18	155.63	0	66, 69, 87	438	41.5972	12.2218	
00045D68-ED2E-4F2F-8186-CEE074D875D0	CcS-6699	b-2390	2020-07-14 15:37:45	326.01	0	30, 11, 16, 81	2118	29.7573	-95.3796	
000481C3-1C26-4FEF-83A0-4CD0EB0048BD	CcS-6696	b-2230	2017-09-04 19:44:53	161.60	0	72	2115	53.5489	-113.503	
00051AA4-9CBE-4268-B070-C38062A1B3E2	CcS-7606	b-2266	2017-01-05 18:19:25	148.91	0	18	3025	52.2084	5.69081	
0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	CcU-3358	b-2598	2023-09-23 04:51:43	294.59	0	35, 33, 19	215	53.5535	-113.499	
0009A151-9BCF-4E31-9053-A468FF77FAAB	CcS-7509	b-2546	2023-12-31 00:06:36	383.63	0	93, 55, 28, 91	2928	51.9362	5.34265	
0009D494-6245-4DF9-955D-2C084191CFFB	CcS-8483	b-2526	2017-07-18 07:52:02	197.80	0	55, 8, 72	3902	45.492	-73.5706	
000A1DEC-CDB6-4AB2-A619-71DA88D4A262	CcS-6467	b-2558	2018-09-08 05:29:58	339.94	0	46, 56, 73	1886	55.7425	-3.30009	
000A1E64-1414-40B0-9D92-5678A4D958E2	CcS-5966	b-2550	2022-09-17 04:02:19	369.71	0	6, 89, 19	1385	52.0821	5.28424	
000A5879-3472-41D9-AF60-42D35038543C	CcU-4569	b-2590	2020-02-07 23:03:45	162.43	0	58, 61, 55	42	39.949	-75.1719	
000AE0D4-F06E-4146-804A-5A2FC73110D7	CcS-8134	b-2426	2017-07-10 07:11:41	188.94	0	62	3553	46.8055	-71.2149	
000BCBF8-A4AB-4E8E-A148-2E7A04E1B385	CcS-6020	b-2598	2019-05-28 04:23:02	96.22	0	36	1439	33.4472	-112.072	

transactions 1 x

Output

Action Output

#	Time	Action	Message
35	14:50:31	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions....'	100000 row(s) affected Records: 100000 Deleted: 0 Skipped: 0 Warnings: 0
36	14:50:43	SELECT * FROM transactions.transactions	100000 row(s) returned

A continuación, se crean las relaciones entre las tablas para crear el modelo estrella, en el cual la tabla transactions está en el centro como la tabla de hechos y el resto de las tablas son las dimensiones.

Después de estudiar cuidadosamente las tablas al cargar los datos, se evidencia que las tablas american\_users y european\_users se componen de los mismos campos y los id se complementan entre sí, es decir, que si se unen las dos tablas no habría problema de id repetidos y se ahorra espacio de una tabla. No estoy segura de cuál sea la mejor práctica, pero opté por crear una nueva tabla llamada simplemente users que se compone de la unión de las tablas american\_users y european\_users; a partir de aquí se pretende trabajar únicamente con esta tabla users. Antes de realizar la unión de estas dos tablas nos aseguramos de que no haya id repetidos entre ambas tablas, para ello se realizó un JOIN entre ellas, para confirmar que no hay solapamientos, como se muestra a continuación:

```

154 • SELECT a.id AS americans, e.id AS europeans
155 FROM american_users AS a
156 JOIN european_users AS e
157 ON a.id=e.id;
158

```

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

americans	europeans
-----------	-----------

Result 6 x

Output

Action Output

#	Time	Action	Message
✓ 24	12:28:13	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_us...	3990 row(s) affected Records: 3990 Deleted: 0 Skipped: 0 Warnings: 0
✓ 25	12:29:36	SELECT a.id AS americans, e.id AS europeans FROM american_users AS a JOIN europe...	0 row(s) returned

Al realizar esta unión, se podría añadir una columna que diferencie cada id si es de Europa o de america, pero como ya se tiene una columna con el país no se considera que sea necesaria esa columna adicional para esta instancia, nuevamente, no se tiene conocimiento de las mejores prácticas, pero por ahora no se añade la columna identificativa de continente, si más adelante es necesaria, se evidenciará y se mostrará el proceso por el cual se obtuvo. Para la obtención de la nueva tabla users se empleó la unión que se muestra a continuación en la captura de pantalla:

```

159 # union tabla european_users y tabla european_users:
160 • CREATE TABLE IF NOT EXISTS users(
161 SELECT id,name,surname,phone,email,birth_date,country,city,postal_code,address
162 FROM american_users
163 UNION
164 SELECT id,name,surname,phone,email,birth_date,country,city,postal_code,address
165 FROM european_users);
166

```

Output

Action Output

#	Time	Action	Message
✓ 3	18:50:58	CREATE TABLE IF NOT EXISTS users( SELECT id,name,surname,phone,email,birth_dat...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
✓ 4	18:51:09	SELECT * FROM transactions.users	5000 row(s) returned

```
1 • SELECT * FROM transactions.users;
```

Result Grid										
Filter Rows: <input type="text"/>										
Export:  Wrap Cell Content:  Fetch rows:										
id	name	surname	phone	email	birth_date	country	city	postal_code	address	
145	Ursula	Stewart	1-442-838-6756	commodo.auctor.velit@outlook.ca	Feb 17, 1994	United States	Houston	77001	161-6225 Ac	
146	Priscilla	Skinner	(468) 855-0771	laoreet.lectus@aol.edu	Jul 31, 1980	United States	Philadelphia	19101	207-6998 At Ave	
147	Brody	Talley	(307) 307-2751	metus.sit.amet@outlook.org	Jun 13, 1991	United States	Houston	77001	469-5852 Tellus Street	
148	Kerry	Adkins	(528) 872-1974	augue.eu.tempor@icloud.couk	Mar 13, 1983	United States	Chicago	60601	Ap #422-4836 Nunc Rd.	
149	Brock	Doyle	(265) 140-9567	curtus.a@aol.edu	Feb 19, 1986	United States	Chicago	60601	Ap #172-5737 Lorem St.	
150	Oleg	Coleman	1-131-139-5673	dis@outlook.edu	Dec 2, 1981	United States	Chicago	60601	722-3056 Eu	
151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	Jul 2, 1980	United Kingdom	London	EC1A 1BB	Ap #432-4493 Aliquet Rd.	
152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google.edu	Sep 30, 1979	United Kingdom	Birmingham	B1 1AA	551-8930 Lobortis Street	
153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	London	EC1A 1BB	Ap #312-5898 Consectet...	
154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Manchester	M1 1AE	872-1866 Pede Rd.	
155	Joshua	Russell	055 4409 5286	justo.nec.ante@outlook.edu	Jan 23, 1984	United Kingdom	Manchester	M1 1AE	Ap #285-4727 Auctor. Av.	
156	Remedios	Case	055 3114 1566	mollis.phasellus.libero@aol.com	Oct 9, 1994	United Kingdom	Liverpool	L1 1AA	479-3690 Turpis Road	
157	Philip	Carey	0800 640 6251	phasellus@yahoo.net	Oct 10, 1992	United Kingdom	Manchester	M1 1AE	196-1103 Quisque Street	
158	Fatima	Dyer	0800 1111	adipiscing@google.org	Dec 24, 1988	United Kingdom	Birmingham	B1 1AA	Ap #435-7194 Scelerisqu...	
159	Edmund	Armstrong	066 5737 6623	dispartia.lorem.aliquam@protonmail.com	May 18, 2000	United Kingdom	Liverpool	L1 1AA	871-3506 Lobortis Ave	

#	Time	Action	Message
3	18:50:58	CREATE TABLE IF NOT EXISTS users( SELECT id,name,sumame.phone,email,birth_dat...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0
4	18:51:09	SELECT * FROM transactions.users	5000 row(s) returned

Una vez realizada la unión se eliminan las tablas individuales pues ya no son necesarias y así no se tienen datos repetidos, así (ignorar el comentario de las fk que aparece):

```
171 • DROP TABLE american_users;
172 • DROP TABLE european_users;
173
```

Output			
Action Output			
#	Time	Action	Message
1	22:33:50	SELECT * FROM transactions.transactions	100000 row(s) returned
2	22:34:13	DROP TABLE american_users	0 row(s) affected
3	22:34:19	DROP TABLE european_users	0 row(s) affected

Considerando que la tabla transacciones es la tabla de hechos y por ende el centro del modelo estrella, todas las fk se hacen con base en esta. Por lo que a continuación se muestran las capturas de pantalla con las líneas de código para la creación de la relación de cada una de las tablas con la tabla transactions y posteriormente se dará una breve síntesis del modelo resultante.

- Relación transactions-users:

En primera instancia, se generó un error, luego me di cuenta de que era porque al crear la tabla users no se indicó cuál era la pk, por ende, al hacer la relación fk en transactions no la identificaba. En la captura de pantalla a continuación se muestra la query correspondiente para establecer la relación y el error generado en los outputs, las líneas de código 175 y 176 que se emplearon para solventar el error.

```

167 #transactions- users:
168 • ALTER TABLE transactions
169 ADD CONSTRAINT FK_t_u
170 FOREIGN KEY (user_id)
171 REFERENCES users(id);
172 #me da error 1822 el mismo error dde la tabla padre
173 #Vale depronto sea porque al crear la tabla users con la union de las otras dos no le indique cual era la pk de esa nueva tabla users
174
175 • ALTER TABLE users
176 ADD CONSTRAINT users_pk PRIMARY KEY (id);
177

```

Output

#	Time	Action	Message	Du
✓ 12	19:17:39	SELECT * FROM transactions.users	5000 row(s) returned	0.0
✗ 13	19:18:45	ALTER TABLE transactions ADD CONSTRAINT FK_t_u FOREIGN KEY (user_id) REFE...	Error Code: 1822. Failed to add the foreign key constraint. Missing index for constraint 'FK...	0.0
✓ 14	19:25:39	ALTER TABLE users ADD CONSTRAINT users_pk PRIMARY KEY (id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	1.6
✓ 15	19:25:47	ALTER TABLE transactions ADD CONSTRAINT FK_t_u FOREIGN KEY (user_id) REFE...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0	8.5

En la siguiente captura de pantalla se muestra los queries necesarios para la creación de las relaciones mediante fk, primero entre la tabla transactions y companies, y posteriormente entre la tabla transactions y credit\_cards:

```

176 #transactions- companies:
177 • ALTER TABLE transactions
178 ADD CONSTRAINT FK_t_c
179 FOREIGN KEY (business_id)
180 REFERENCES companies(company_id);
181
182 #transactions- credit_cards:
183 • ALTER TABLE transactions
184 ADD CONSTRAINT FK_t_cc
185 FOREIGN KEY (card_id)
186 REFERENCES credit_cards(id);
187

```

Output

#	Time	Action	Message
✓ 12	11:14:28	ALTER TABLE transactions ADD CONSTRAINT FK_t_c FOREIGN KEY (business_id) R...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
✓ 13	11:14:55	ALTER TABLE transactions ADD CONSTRAINT FK_t_cc FOREIGN KEY (card_id) REFE...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

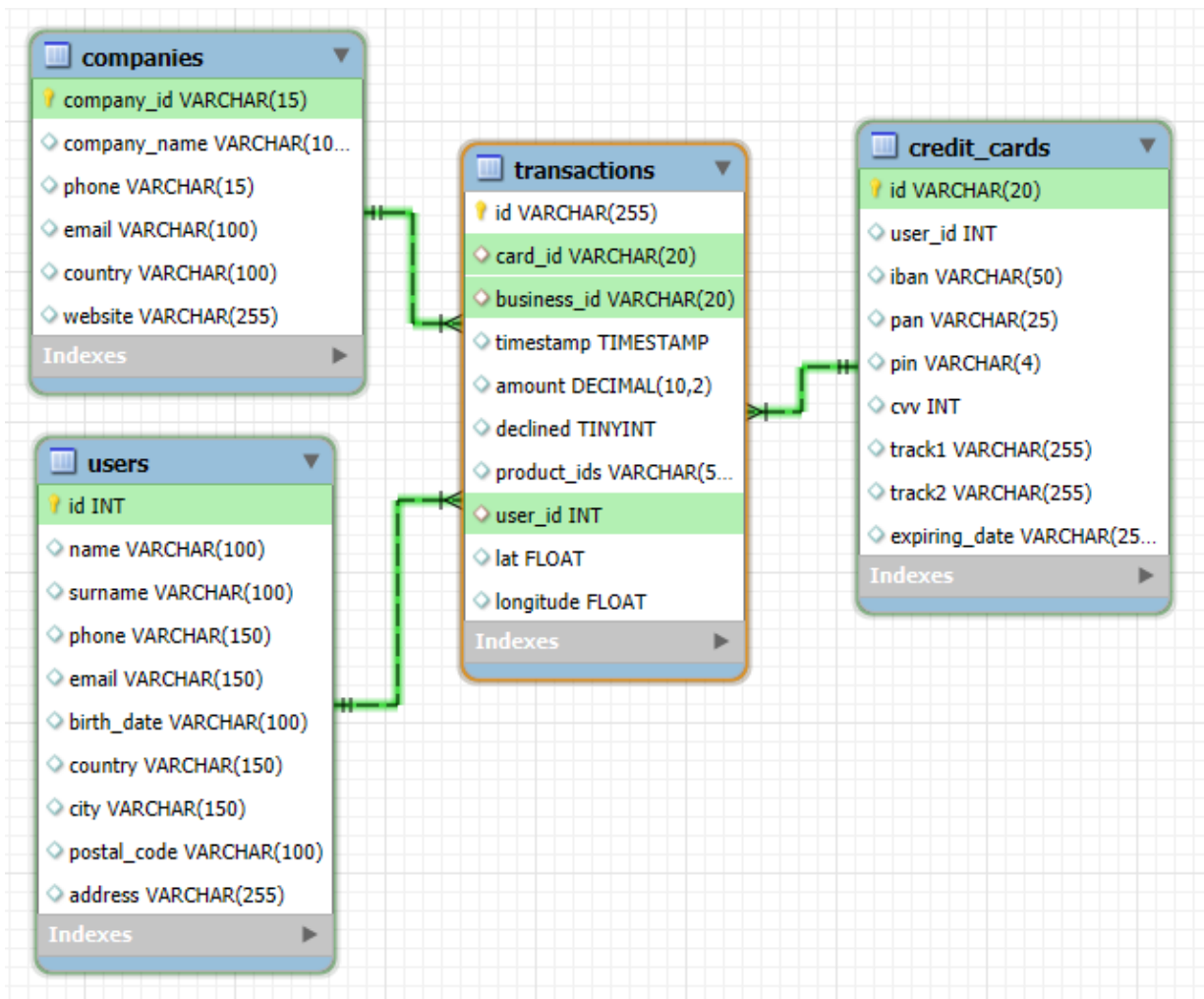
Con base a la corrección, tanto del Sprint 3 como del compañero con quien se realizó el p2p; la construcción de las forein keys se puede hacer en una única query con lo que se ahorra líneas de código y es menos confuso, además de corresponder con buenas prácticas, de la siguiente forma:

```

178 • ALTER TABLE transactions
179     ADD CONSTRAINT FK_t_u FOREIGN KEY (user_id) REFERENCES users(id), #con users
180     ADD CONSTRAINT FK_t_c FOREIGN KEY (business_id) REFERENCES companies(company_id), #con companies
181     ADD CONSTRAINT FK_t_cc FOREIGN KEY (card_id) REFERENCES credit_cards(id) #con credit caards
182     ;

```

Esquema de estrella que contenga, al menos 4 tablas obtenido:



### Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 80 transacciones utilizando al menos 2 tablas.

```

206      #N1
207      #Ejercicio 1:
208      #usuarios con más de 80 transacciones
209 •   SELECT u.name, u.surname, COUNT(t.id) AS num_transacciones
210     FROM transactions AS t
211    JOIN users AS u
212     ON t.user_id=u.id
213    GROUP BY u.name, u.surname
214    HAVING COUNT(t.id) > 80;
215

```

Result Grid

	name	surname	num_transacciones
▶	Molly	Gilliam	110
	Dxwgi	Hwcru	94
	Bnyr	Astuw	91
	Sfzzoh	Xgvfridxs	81

Result 3 x

Output

Action Output

#	Time	Action	Message
✓ 22	12:47:35	SELECT u.name, u.sumame, COUNT(t.id) AS num_transacciones FROM transactions AS...	4 row(s) returned
✓ 23	12:49:56	SELECT * FROM transactions.credit_cards	5000 row(s) returned

Aquí por un pequeño despiste, lo hice con JOIN afortunadamente los p2p son para estos detalles y a continuación se muestra cómo se realizó con subqueries:

```

222 •   SELECT u. id, u.name, u.surname
223     FROM users AS u
224    WHERE u.id IN (SELECT t.user_id
225                  FROM transactions AS t
226                 GROUP BY t.user_id
227                 HAVING COUNT(t.id) > 80)
228    ;

```

Result Grid

	id	name	surname
▶	185	Molly	Gilliam
	289	Dxwgi	Hwcru
	318	Bnyr	Astuw
	454	Sfzzoh	Xgvfridxs
*	NULL	NULL	NULL

users 5 x

Output

Action Output

#	Time	Action	Message
✗ 23	23:05:26	SELECT u. id, u.name, u.sumame, COUNT(t.id) FROM users AS u WHERE (SELECT t.us...	Error Code: 1054. Unknown column 't.id' in 'field list'
✗ 24	23:05:35	SELECT u. id, u.name, u.sumame FROM users AS u WHERE (SELECT t.user_id FROM t...	Error Code: 1242. Subquery returns more than 1 row
✗ 25	23:06:54	SELECT u. id, u.name, u.sumame FROM users AS u WHERE u.id= (SELECT t.user_id F...	Error Code: 1242. Subquery returns more than 1 row
✗ 26	23:07:00	SELECT u. id, u.name, u.sumame FROM users AS u WHERE u.id= (SELECT t.user_id F...	Error Code: 1242. Subquery returns more than 1 row
✓ 27	23:07:25	SELECT u. id, u.name, u.sumame FROM users AS u WHERE u.id IN (SELECT t.user_id ...	4 row(s) returned



Por estética es que yo preferiría que se presentara el conteo de transacciones en la tabla resultante, a pesar de que con la subquery en cláusula WHERE se obtiene la respuesta que solicita la consulta, no se puede mostrar esta columna. por capricho lo intenté de la siguiente forma:

```
229 • SELECT u.id, u.name, u.surname,  
230      (SELECT COUNT(t.id)  
231        FROM transactions AS t  
232        WHERE u.id= t.user_id) AS conteo_transacciones  
233 FROM users AS u  
234 HAVING conteo_transacciones >80;  
235  
236  
237
```

Result Grid

id	name	surname	conteo_transacciones
185	Molly	Gilliam	110
289	Dxwgi	Hwcru	94
318	Bnyr	Astuw	91
454	Sfzzoh	Xgvfridxs	81

Result 6 x

Output

Action Output

#	Time	Action	Message
28	23:55:06	SELECT u.id, u.name, u.username, (SELECT COUNT(t.id) FROM transactions AS t	Error Code: 1054. Unknown column 'u.username' in 'field list'
29	23:55:30	SELECT u.id, u.name, u.surname, (SELECT COUNT(t.id) FROM transactions AS t	4 row(s) returned

Suelo hacer las consultas siempre por el WHERE más que todo porque se me facilita el entendimiento en mi cabeza, en este punto me costó un poco entender la lógica de la subconsulta desde la SELECT pero es bueno haber practicado.

## **N1- Ejercicio 2:**

Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.

```

223 • SELECT cc.id AS id_tarjeta, cc.iban AS iban_terjeta,
224         ROUND(AVG(t.amount),2) AS media_cantidad,
225         c.company_name AS nombre_empresa
226 FROM transactions AS t
227 JOIN credit_cards AS cc
228     ON t.card_id=cc.id
229 JOIN companies AS c
230     ON t.business_id=c.company_id
231 WHERE c.company_name='Donec Ltd'
232 GROUP BY cc.iban, c.company_name, cc.id;

```

Result Grid

	id_tarjeta	iban_terjeta	media_cantidad	nombre_empresa
▶	CcS-5928	XX911406401125586307586805	356.25	Donec Ltd
	CcU-4009	SK9446370242474562577506	142.96	Donec Ltd
	CcS-6066	XX776752917845952975555640	257.37	Donec Ltd
	CcS-8452	XX413827362289719304908990	139.59	Donec Ltd
	CcS-7810	XX347787246070769610780308	240.41	Donec Ltd
	CcS-6547	XX688768436543090894854602	188.58	Donec Ltd
	CcL-4135	MX78368851538688340	430.30	Donec Ltd

Result 4 x

Output

Action Output

#	Time	Action	Message
✓ 10	10:18:09	SELECT cc.id AS id_tarjeta, cc.iban AS iban_terjetas, ROUND(AVG(t.amount),2) AS medi...	371 row(s) returned
✓ 11	10:23:02	SELECT cc.id AS id_tarjeta, cc.iban AS iban_terjeta, ROUND(AVG(t.amount),2) AS media...	371 row(s) returned

## Nivel 2:

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basada en si las tres últimas transacciones han sido declinadas, entonces es inactivo; si al menos una no es rechazada, entonces es activo.

Inicialmente se creó la nueva tabla que se pide en la consulta:

```

240 #creacion de la nueva tabla con las columnas que necesito
241 • CREATE TABLE estado_tarjetas AS
242     SELECT card_id, id AS id_transaction, timestamp, declined
243     FROM transactions;
244

```

Output

Action Output

#	Time	Action	Message
✓ 1	10:36:31	CREATE TABLE estado_tarjetas AS SELECT card_id, id AS id_transaction, timestamp, decline...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0

Una vez creada la tabla, se pretendía llenar con los datos que pedía la consulta; sin embargo, esto no es posible con una query simple, hace falta emplear una función un poco más avanzada de las que se han empleado hasta ahora.

Las funciones de ventana son un tipo de funciones de agregado. Por su parte, las funciones de agregación no muestran tanto detalle como las funciones de ventana; estas últimas permiten la creación de una columna con la operación que se necesita y al mismo tiempo mantener el detalle. En otras palabras: se parte de un conjunto macro, se crean

subconjuntos y las funciones de ventana permiten hacer cálculos con base en esos subconjuntos.

#ROW\_NUMBER enumera las filas, es un superpoder de las funciones de ventana, siempre va con OVER y dentro del OVER siempre va incluido PARTITION y ORDER BY.

```
252 #funcion de ventana para agrupar por card_id y ordenar por timestamp
253 • SELECT *,
254 ROW_NUMBER () OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS ranking_transactions
255 FROM estado_tarjetas
256 ;
```

The screenshot shows a database interface with a SQL query and its results. The query is a window function that ranks transactions by timestamp for each card ID. The results table shows 10 rows of transaction data, with the most recent transaction for each card ID having a ranking of 1. Below the results, the 'Output' section shows the execution of the query, indicating that 100,000 rows were returned.

card_id	id_transaction	timestamp	declined	ranking_transactions
CcS-4857	621CB9D6-7896-401D-BD38-3F8D06470B72	2018-03-02 23:33:26	0	19
CcS-4857	FD97C4C4-F01C-4D9D-AE1E-1171FF302269	2017-12-29 16:19:35	0	20
CcS-4857	54B07074-2BE4-4CA5-901E-A6816D1E2540	2016-11-14 06:00:28	0	21
CcS-4857	26904E68-80E1-41DF-9ECB-3A401C08787E	2016-10-12 12:27:29	0	22
CcS-4858	368F5338-3A43-49F2-9EAE-5FE6268EB4F9	2024-08-28 13:48:22	0	1
CcS-4858	485D254E-4031-4563-80FD-6606793E7185	2023-12-10 10:56:40	0	2
CcS-4858	5B179060-5336-4F81-BDC7-06A0198C0178	2023-12-10 09:21:55	0	3
CcS-4858	63D4FDFC-6060-4F29-94D6-AD7F17081BF0	2023-12-10 02:07:04	0	4
CcS-4858	3C3C634D-EB56-4892-9B46-5C6DC00FC252	2023-12-08 21:23:03	0	5
CcS-4858	D1CE2ACE-5BC6-4332-B75A-4C7EC0B020BB	2023-11-27 07:43:39	0	6
CcS-4858	D413FAB7-61A0-4A9D-B259-2F6027A99EA2	2023-08-02 08:48:27	0	7
CcS-4858	33A6347C-C2E0-49A9-B5FD-3622B51B13D7	2023-03-23 00:56:58	0	8
CcS-4858	EE4B48DA-5991-4AF0-9E8B-53A2A2EE2FD7	2023-01-13 13:07:24	0	9

Result 1 x

Output

Action Output

#	Time	Action	Message
1	10:36:31	CREATE TABLE estado_tarjetas AS SELECT card_id, id AS id_transaction, timestamp, dec...	100000 row(s) affected Records: 100000 Duplicates: 0 Warnings: 0
2	10:46:21	SELECT *, ROW_NUMBER () OVER (PARTITION BY card_id ORDER BY timestamp DES...	100000 row(s) returned

Entonces para obtener las 3 últimas transacciones de cada tarjeta y establecer el estado (activo/inactivo) de cada tarjeta; es necesario emplear las funciones de ventana para agrupar por card\_id y ordenar por timestamp, con esto se obtiene una columna en la que se ranquea cada transacción de cada tarjeta, como se muestra con anterioridad, la enumeración en la columna de ranking se reinicia cuando cambia de un id a otro. Una vez obtenido el ranking, esto permite obtener las tres últimas transacciones de cada tarjeta para establecer el estado de cada una, así:

```

258     #ahora necesito perderle el top 3 de cada card_id
259 •   SELECT *
260     FROM (SELECT *,
261             ROW_NUMBER () OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS ranking_transactions
262             FROM estado_tarjetas) AS ranking_top_3
263     WHERE ranking_top_3.ranking_transactions <= 3;

```

card_id	id_transaction	timestamp	declined	ranking_transactions
CcS-4857	BDD988A9-51B9-4BD8-9E4C-84BB1DE64F5B	2024-10-25 13:11:54	0	1
CcS-4857	B954A0B3-9314-4615-ACBB-B31E0245D8D1	2024-10-07 16:43:17	0	2
CcS-4857	2C537AD9-DB0E-402A-A75F-7D155138F47C	2024-07-27 10:50:49	0	3
CcS-4858	368F5338-3A43-49F2-9EAE-5FE6268EB4F9	2024-08-28 13:48:22	0	1
CcS-4858	485D254E-4031-4563-80FD-6606793E7185	2023-12-10 10:56:40	0	2
CcS-4858	5B179060-5336-4F81-BDC7-06A0198C0178	2023-12-10 09:21:55	0	3
CcS-4859	E365DB9F-B6A9-4E4E-B209-239D0BBEA9E5	2024-06-15 21:04:11	0	1
CcS-4859	95A99862-D57B-4620-9FD7-3AB8F7721D5B	2023-11-11 15:18:56	0	2
CcS-4859	435EF78C-203F-46D3-9EC4-5F18E7537768	2023-03-11 06:20:59	0	3
CcS-4860	835194BB-5404-47A7-B3DB-2960D548E788	2024-10-02 08:31:13	0	1
CcS-4860	08B508AC-5BC9-438A-B29C-8E9C7AD572DA	2024-06-06 18:23:47	0	2

Result 2 x

Output

Action Output

#	Time	Action	Message
✓ 2	10:46:21	SELECT *, ROW_NUMBER () OVER (PARTITION BY card_id ORDER BY timestamp DE...	100000 row(s) returned
✓ 3	10:51:16	SELECT * FROM (SELECT *, ROW_NUMBER () OVER (PARTITION BY card_id ORDE...	15000 row(s) returned

Ahora bien, se pretende evaluar el estado de cada tarjeta teniendo en cuenta que, si las tres últimas transacciones han sido declinadas, entonces es inactivo; si al menos una no es rechazada, entonces es activo. Llevando estas condiciones un poco más al detalle y teniendo en cuenta que la columna declined es binaria:

- declined 0=false=aceptada
- declined 1=true=declinada
- inactiva → ranking=3
- activa → ranking >=1

Para establecer estas condiciones se empleó CASE que permite obtener un valor en función de una o varias condiciones, como se muestra a continuación:

(CASE

WHEN SUM(declined) = 3 THEN 'inactiva'

ELSE 'activa'

END AS estado\_tarjetas );

Aquí es importante considerar las condiciones del CASE, teniendo en cuenta que la columna declined es binaria y la condición va de 0 a 3 pues si la primera condición es igual al máximo que puede ser, cualquier otro valor que resulte, será activa.

Ahora, teniendo en cuenta los pasos anteriores, se realiza una sola consulta, donde se crea la tabla que se solicita en la consulta, se hace uso de la función de ventana y del CASE para obtener el estado de cada una de las tarjetas. A continuación, se muestra la query y la tabla resultante:

```

277 #Vale pues ahora unir todas las partes en una sola query
278 • CREATE TABLE estado_tarjetas AS
279     SELECT card_id,
280            CASE
281                WHEN SUM(declined) = 3 THEN 'inactiva'
282                ELSE 'activa'
283            END AS estado_tarjetas
284     FROM (SELECT card_id, id AS id_transaction, timestamp, declined,
285                ROW_NUMBER () OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS ranking_transactions
286         FROM transactions) AS ranking_top_3
287     WHERE ranking_top_3.ranking_transactions <= 3
288     GROUP BY card_id;
289

```

Output			
Action Output			
#	Time	Action	Message
19	12:13:07	DROP TABLE `transactions`.`estado_tarjetas_2`	0 row(s) affected
20	12:27:45	CREATE TABLE estado_tarjetas AS SELECT card_id, CASE WHEN SUM(declined) = 3 ...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

Sprint 4\*

transactions

credit\_cards

companies

estado\_tarjetas ×

📁

💾

⚡

🔧

🔍

👤

🔗

✅

❌

🔄

Don't Limit

⌵

★

📌

🔍

📄

🔄

```

1 • SELECT * FROM transactions.estado_tarjetas
2   ORDER BY estado_tarjetas DESC;
3

```

Result Grid

📊

🔗

Filter Rows:

Export:

📄

Wrap Cell Content:

🔗

Fetch rows:

🔄

	card_id	estado_tarjetas
▶	CcS-4998	inactiva
	CcS-4870	inactiva
	CcS-5035	inactiva
	CcS-4899	inactiva
	CcU-3568	inactiva
	CcS-4858	activa
	CcS-4859	activa
	CcS-4860	activa
	CcS-4861	activa
	CcS-4862	activa
	CcS-4863	activa
	CcS-4864	activa
	CcS-4865	activa

estado\_tarjetas 3 ×

Output

📄

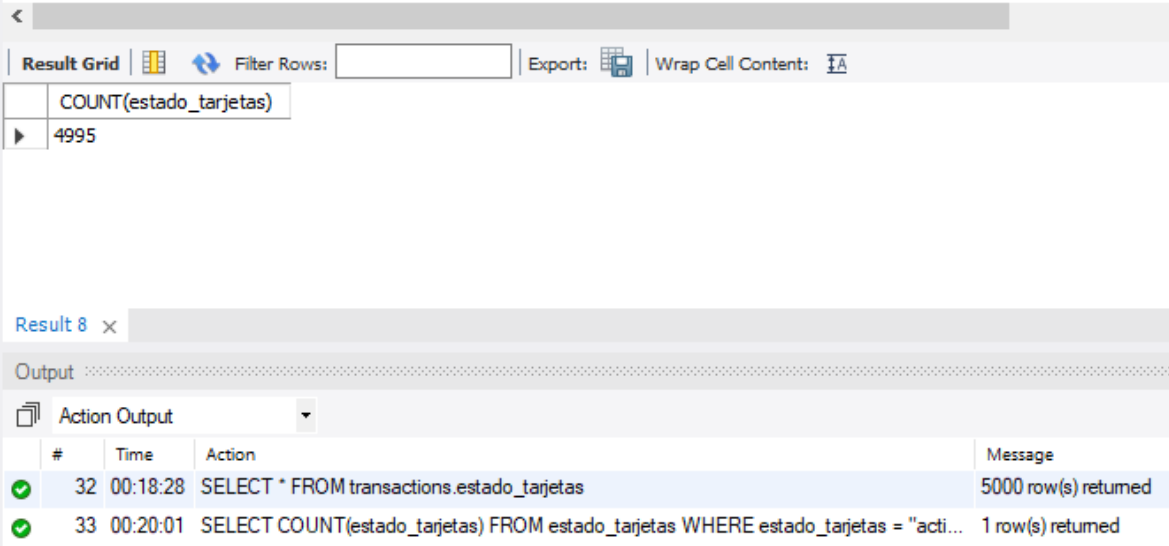
Action Output

⌵

#	Time	Action	Message
✓ 22	12:30:07	SELECT * FROM transactions.estado_tarjetas ORDER BY estado_tarjetas	5000 row(s) returned
✓ 23	12:30:12	SELECT * FROM transactions.estado_tarjetas ORDER BY estado_tarjetas DESC	5000 row(s) returned

¿Cuántas tarjetas están activas?

```
311 #Cuantas estan activas?
312 • SELECT COUNT(estado_tarjetas)
313 FROM estado_tarjetas
314 WHERE estado_tarjetas = "activa";
315
```



The screenshot shows a database interface with a query result grid. The query is: `SELECT COUNT(estado_tarjetas) FROM estado_tarjetas WHERE estado_tarjetas = "activa";`. The result grid shows a single row with the value 4995. Below the grid, there is an 'Output' section with a table of actions and messages.

#	Time	Action	Message
32	00:18:28	SELECT * FROM transactions.estado_tarjetas	5000 row(s) returned
33	00:20:01	SELECT COUNT(estado_tarjetas) FROM estado_tarjetas WHERE estado_tarjetas = "activa";	1 row(s) returned

En total hay 4995 tarjetas activas, es decir, que las inactivas son solo 5.

### 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 product\_ids. Genera la siguiente consulta:

#### Ejercicio 1:

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

Después de la aclaración de franco (alabado sea franco):

Para unir la tabla productos con la tabla transacciones hay que tener en cuenta que se deberían unir a través de la columna id de la tabla productos con la columna products\_ids de la tabla transacciones. Pero aquí empieza a oscurecerse el asunto; Esta última columna de la tabla transactions, contiene más de un valor en un mismo campo, lo cual genera una relación N:N entre estas dos tablas. Cosa que debe de evitarse a toda costa en este tipo de modelos.

Para aclarar este asunto, se hace uso de una tabla puente, la cual permite la unión de estas dos tablas por medio de relaciones N:1 desde la tabla puente hacia cada una de las tablas involucradas (products y transactions). Esta tabla puente se caracteriza por estar formada por una primary key compuesta, es decir, un conjunto de dos columnas (id\_transactions y id\_productos) para así unirse con las otras dos tablas.

Para lo anterior, se crea la tabla puente como se muestra a continuación:

```

289
290 #Creacion de la tabla puente:
291 CREATE TABLE tabla_puente (
292     transaction_id VARCHAR(100),
293     product_id INT,
294     PRIMARY KEY (transaction_id, product_id),
295     FOREIGN KEY (transaction_id) REFERENCES transactions.transactions(id),
296     FOREIGN KEY (product_id) REFERENCES transactions.products(id)
297 );

```

Output				
Action Output				
#	Time	Action	Message	
✓ 42	12:58:17	SELECT id AS id_transactions, product_ids, CONCAT('[', REPLACE(t.product_ids, ',', ' '), '], [', ...	100000 row(s) returned	
✓ 43	12:58:22	CREATE TABLE tabla_puente ( transaction_id VARCHAR(100), product_id INT, PRI...	0 row(s) affected	

Ahora bien, respecto al tema de que hay más de un dato por campo en la columna products\_ids, es necesario emplear herramientas mucho más avanzadas con el propósito de “separar” cada uno de estos ids, sin perder información. Para esto se emplea el formato JSON, el cual permite precisamente lo anterior, a continuación, se muestra la diferencia de formatos una vez se pasa la columna product\_ids a un formato JSON:

```

290 #Nivel 3
291 #Ejercicio 1
292 #product_ids convertida a JSON
293 SELECT id AS id_transactions, product_ids, CONCAT('[', REPLACE(t.product_ids, ',', ' '), '], [', ... AS t_prod_ids_json
294 FROM transactions AS t;

```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
	id_transactions	product_ids	t_prod_ids_json				
▶	00043A49-2949-494B-A5DD-A5BAE3B819DD	16, 26, 97, 87	[16],[ 26],[ 97],[ 87]				
	000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	66, 69, 87	[66],[ 69],[ 87]				
	00045D6B-ED2E-4F2F-8186-CEE074D875D0	30, 11, 16, 81	[30],[ 11],[ 16],[ 81]				
	000481C3-1C26-4FEF-83A0-4CD0EB004BBD	72	[72]				
	00051AA4-9CBE-4268-B070-C38062A1B3E2	18	[18]				
	0008A312-EDFE-4A4F-BC99-E9C92EC3CA4D	35, 33, 19	[35],[ 33],[ 19]				
	0009A151-9BCE-4E31-9053-A468FF77FAAB	93, 55, 28, 91	[93],[ 55],[ 28],[ 91]				
	0009D494-6245-4DF9-955D-2C084191CFFB	55, 8, 72	[55],[ 8],[ 72]				
	000A1DEC-CDB6-4AB2-A619-71DAB8D4A262	46, 56, 73	[46],[ 56],[ 73]				
	000A1E64-1414-40B0-9D92-5678A4D958E2	6, 89, 19	[6],[ 89],[ 19]				
	000A5879-3472-41D9-AF60-42D3503B543C	58, 61, 55	[58],[ 61],[ 55]				
	000AE0D4-F06E-4146-804A-5A2FC73110D7	62	[62]				
	000BCBF8-A4AB-4E8E-A148-2E7A04E1B385	36	[36]				

Result 4 x

Output				
Action Output				
#	Time	Action	Message	
✗ 17	14:52:18	JSON_TABLE (CONCAT('[', REPLACE(transactions.product_ids, ',', ' '), '], [', ...	Error Code: 1064. You have an error in your SQL syntax; check the r	
✓ 18	15:00:10	SELECT id AS id_transactions, product_ids, CONCAT('[', REPLACE(t.product_ids, ',', ' '), '], [', ...	100000 row(s) returned	

Sin embargo, lo anterior, es solo un SELECT, por lo que los resultados mostrados no están “guardados”, por así decirlo, en ninguna parte por lo que para indicar que queremos esto en un objeto para su posterior uso se emplea JSON\_TABLE, que a su vez convierte estos datos en columnas (traspone) y posteriormente se insertan estos datos en la tabla puente creada con anterioridad, de la siguiente forma:

```

303 • SELECT id AS id_transactions,
304         product_ids,
305         CONCAT('[', REPLACE(product_ids, ',', '],[', '],') AS t_prod_ids_json
306 FROM transactions AS t;
307
308 • INSERT INTO tabla_puente (transaction_id, product_id)
309 SELECT
310     t.id AS transaction_id,
311     jt.product_id
312 FROM transactions AS t,
313 JSON_TABLE(
314     CONCAT('[', t.product_ids, ']'),
315     '$[*]' COLUMNS (product_id INT PATH '$')
316 ) AS jt;

```

Output			
Action Output			
#	Time	Action	Message
✓ 48	13:16:28	DROP TABLE `transactions`.`tabla_puente`	0 row(s) affected
✓ 49	13:16:52	CREATE TABLE `tabla_puente` ( `transaction_id` VARCHAR(100), `product_id` INT, PRIMARY KEY (`transaction_id`, `product_id`))	0 row(s) affected
✗ 50	13:16:55	INSERT INTO `tabla_puente` (`transaction_id`, `product_id`) SELECT `t`.`id` AS `transaction_id`, ...	Error Code: 3141. Invalid JSON text in argument 1 to function json_table: "The document ...
✓ 51	13:17:47	INSERT INTO `tabla_puente` (`transaction_id`, `product_id`) SELECT `t`.`id` AS `transaction_id`, ...	253391 row(s) affected Records: 253391 Duplicates: 0 Warnings: 0
✓ 52	13:18:12	SELECT * FROM `transactions`.`tabla_puente`	253391 row(s) returned

Una vez introducidos los datos en la tabla puente, se obtiene algo así:

```

1 • SELECT * FROM transactions.tabla_puente;

```

transaction_id	product_id
001A60EA-DC9C-4E5A-9460-6628B100E7E1	1
0032F0BB-BBE6-4AA5-B5EE-EEAD533C0C48	1
00342381-503D-422D-85AB-F2D4FFAAD4C7	1
004C0A80-E537-46D8-BE44-343D2176DF15	1
004D1DB5-B2CB-4460-98B6-31C42CA96E5F	1
0062599C-0A1F-4405-AD55-45E20043B551	1
007B1297-C3ED-4966-802C-68FE106AD25C	1
007EA15D-AA1F-4FB3-9483-7D055E99CEC0	1
009679D1-1849-47C1-98E8-8F6C0EA5DF32	1
0096A25B-BDAD-47DE-85AD-BA3741BE7736	1
00C61197-05FF-4ECD-A523-E0D6E2E30DDD	1
00E800C8-F09E-4662-88A4-6D16DF32F0E5	1
00E9A7E4-9FA1-42EC-85BC-ECBE44AAE558	1
00EAF316-6667-4A1F-8CA6-53EDF7EDDF6E	1
00F45CC2-960C-42FF-B4AD-0F4DCC0580BB	1

tabla\_puente 1 x

Output			
Action Output			
#	Time	Action	Message
✗ 1	13:36:42	INSERT INTO `tabla_puente` (`transaction_id`, `product_id`) SELECT `t`.`id` AS `transaction_id`, ...	Error Code: 1062. Duplica
✓ 2	13:36:56	SELECT * FROM `transactions`.`tabla_puente`	253391 row(s) returned

Ahora bien, se necesita saber el número de veces que se ha vendido cada producto, para ello se emplea la siguiente subconsulta y se obtiene el resultado mostrado a continuación:



```

347 #cuantas veces se ha vendido cada producto:
348 • SELECT product_id, COUNT(transaction_id) AS num_ventas_por_producto
349 FROM tabla_puente
350 GROUP BY product_id
351 ORDER BY num_ventas_por_producto DESC;

```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	product_id	num_ventas_por_producto			
▶	52	2654			
	29	2635			
	21	2609			
	16	2608			
	66	2601			
	87	2598			
	33	2597			
	48	2597			
	23	2593			
	68	2589			
	88	2587			
	4	2584			
	73	2584			
	28	2584			
	34	2580			

Result 2 x

Output

Action Output

#	Time	Action	Message
✓ 10	13:46:23	SELECT * FROM transactions.products	100 row(s) returned
✓ 11	13:47:00	SELECT product_id, COUNT(transaction_id) AS num_ventas_por_producto FROM tabla_...	100 row(s) returned

El ORDER BY no era necesario, sin embargo, permite conocer a simple vista cuáles fueron los productos más vendidos.