

Tasca S4.01. Creació de Base de Dades

Nivel 1

Descarga los archivos CSV, estúdielas y diseña una base de datos con un esquema de estrella que contenga, al menos 4 tablas de las que puedas realizar las siguientes consultas: Creamos primero la base de datos transacciones donde estarán todas las tablas que cargaremos desde los archivos CSV's.

Mediante use transacciones; pondremos por defecto esta base de datos.

Descargamos en nuestro ordenador los archivos CSV's y los visualizamos con Visual Studio Code.

Creamos las tablas con Create table y verificamos si existen con el "IF NOT EXISTS". También creamos las Primary key en cada tabla y en las tablas products y users crearemos auto_increment, y tipo de dato INT.

En la mayoría de caso creamos el tipo de dato VARCHAR (255), y en casos que observemos que son enteros: INT, en Price y amount: decimal (10,2), ya que manejaran datos de precios y montos, y en la columna timestamp tipo de dato TIMESTAMP, en las columnas de lat y longitude: FLOAT, ya que son coordenadas.

```
4 • CREATE DATABASE IF NOT EXISTS transacciones;
5 • USE transacciones;
6
7 -- creamos las tablas
8 • CREATE TABLE IF NOT EXISTS companies (
9     company_id VARCHAR(255) PRIMARY KEY,
10    company_name VARCHAR(255),
11    phone VARCHAR(255),
12    email VARCHAR(255),
13    country VARCHAR(255),
14    website VARCHAR(255)
15 );
16
17 • CREATE TABLE IF NOT EXISTS credit_cards (
18     id VARCHAR(255) PRIMARY KEY,
19     user_id INT,
20     iban VARCHAR(255),
21     pan VARCHAR(255),
22     pin VARCHAR(255),
23     cvv INT,
24     track1 VARCHAR(255),
25     track2 VARCHAR(255),
26     expiring_date VARCHAR(255)
27 );
28
29 • CREATE TABLE IF NOT EXISTS products (
30     id INT PRIMARY KEY AUTO_INCREMENT,
31     product_name VARCHAR(255),
32     price DECIMAL(10,2),
33     colour VARCHAR(255),
34     weight VARCHAR(255),
35     warehouse_id VARCHAR(255)
36 );
37
```

Output

Action Output

#	Time	Action	Message
✓ 4	19:47:41	CREATE DATABASE IF NOT EXISTS transacciones	1 row(s) affected
✓ 5	19:47:41	USE transacciones	0 row(s) affected
✓ 6	19:47:41	CREATE TABLE IF NOT EXISTS companies (company_id VARCHAR(255) PRIMARY KEY, company_name VA...	0 row(s) affected
✓ 7	19:47:41	CREATE TABLE IF NOT EXISTS credit_cards (id VARCHAR(255) PRIMARY KEY, user_id INT, iban VARCH...	0 row(s) affected
✓ 8	19:47:41	CREATE TABLE IF NOT EXISTS products (id INT PRIMARY KEY AUTO_INCREMENT, product_name VARCH...	0 row(s) affected

```

37
38 CREATE TABLE IF NOT EXISTS transactions (
39     id VARCHAR(255) PRIMARY KEY,
40     card_id VARCHAR(255),
41     business_id VARCHAR(255),
42     timestamp TIMESTAMP,
43     amount DECIMAL(10, 2),
44     declined TINYINT,
45     product_ids INT,
46     user_id INT,
47     lat FLOAT,
48     longitude FLOAT
49 );
50
51
52 -- los campos de las tablas user_ca, user_uk, users_usa son iguales en estructura, así que uniremos los 3 archivos csv en una sola tabla users
53 CREATE TABLE IF NOT EXISTS users (
54     id INT PRIMARY KEY AUTO_INCREMENT,
55     name VARCHAR(255),
56     surname VARCHAR(255),
57     phone VARCHAR(255),
58     email VARCHAR(255),
59     birth_date VARCHAR(255),
60     country VARCHAR(255),
61     city VARCHAR(255),
62     postal_code VARCHAR(255),
63     address VARCHAR(255)
64 );
65
66 -- Ejercicio 1
67 -- Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.
68

```

Output

Action Output

#	Time	Action	Message
1	20:11:18	CREATE TABLE IF NOT EXISTS transactions (id VARCHAR(255) PRIMARY KEY, card_id VARCHAR(255), b...	0 row(s) affected
2	20:11:18	CREATE TABLE IF NOT EXISTS users (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255), surname...	0 row(s) affected

Comprobamos que se hayan creado las tablas con describe:

- **Tabla company**

```

66 DESC companies;
67

```

Result Grid

Field	Type	Null	Key	Default	Extra
company_id	varchar(255)	NO	PRI	NULL	
company_name	varchar(255)	YES		NULL	
phone	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
country	varchar(255)	YES		NULL	
website	varchar(255)	YES		NULL	

Result 37

Output

Action Output

#	Time	Action	Message
1	18:30:47	DESC companies	6 row(s) returned

• Tabla credit_cards

```
66 • DESC credit_cards;
67
```

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(255)	NO	PRI	NULL	
	user_id	int	YES		NULL	
	iban	varchar(255)	YES		NULL	
	pan	varchar(255)	YES		NULL	
	pin	varchar(255)	YES		NULL	
	cvv	int	YES		NULL	
	track1	varchar(255)	YES		NULL	
	track2	varchar(255)	YES		NULL	
	expiring_date	varchar(255)	YES		NULL	

Result 39 ×

Output

Action Output

#	Time	Action	Message
✓ 1	18:33:36	DESC credit_cards	9 row(s) returned

• Tabla products

```
66 • DESC products;
67
```

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	product_name	varchar(255)	YES		NULL	
	price	varchar(255)	YES		NULL	
	colour	varchar(255)	YES		NULL	
	weight	varchar(255)	YES		NULL	
	warehouse_id	varchar(255)	YES		NULL	

Result 41 ×

Output

Action Output

#	Time	Action	Message
✓ 1	18:35:55	DESC products	6 row(s) returned

- **Tabla transactions**

66 • `DESC transactions;`

67

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(255)	NO	PRI	NULL	
	card_id	varchar(255)	YES		NULL	
	business_id	varchar(255)	YES		NULL	
	timestamp	timestamp	YES		NULL	
	amount	decimal(10,2)	YES		NULL	
	declined	tinyint	YES		NULL	
	product_ids	varchar(255)	YES		NULL	
	user_id	int	YES		NULL	
	lat	float	YES		NULL	
	longitude	float	YES		NULL	

Result 45 ×

Output

Action Output

#	Time	Action	Message
✓ 1	18:38:34	DESC transactions	10 row(s) returned

- **Tabla users**

66 • `DESC users;`

67

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	name	varchar(255)	YES		NULL	
	surname	varchar(255)	YES		NULL	
	phone	varchar(255)	YES		NULL	
	email	varchar(255)	YES		NULL	
	birth_date	varchar(255)	YES		NULL	
	country	varchar(255)	YES		NULL	
	city	varchar(255)	YES		NULL	
	postal_code	varchar(255)	YES		NULL	
	address	varchar(255)	YES		NULL	

Result 46 ×

Output

Action Output

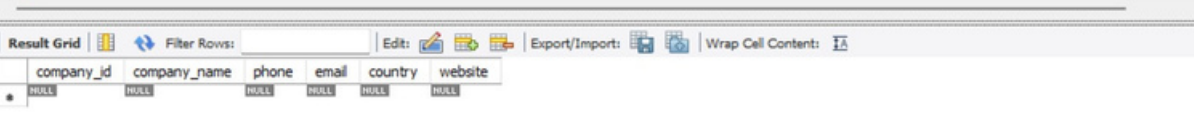
#	Time	Action	Message
✓ 1	18:40:22	DESC users	10 row(s) returned

Introducción de datos en las tablas creadas

- Tabla companies

Previamente a la introducción de datos en las tablas, nos cercioramos de que no haya datos en las tablas, adjunto imagen abajo de una tabla companies, por ejemplo.

```
77 • SELECT * FROM companies; -- miramos si existe información en la tabla companies antes de cargar en introducir los datos.
78
```



The screenshot shows the 'companies' table in SQL Server Enterprise Manager. The table has six columns: company_id, company_name, phone, email, country, and website. All cells in the table are null.

```
companies 48 x
```

Output

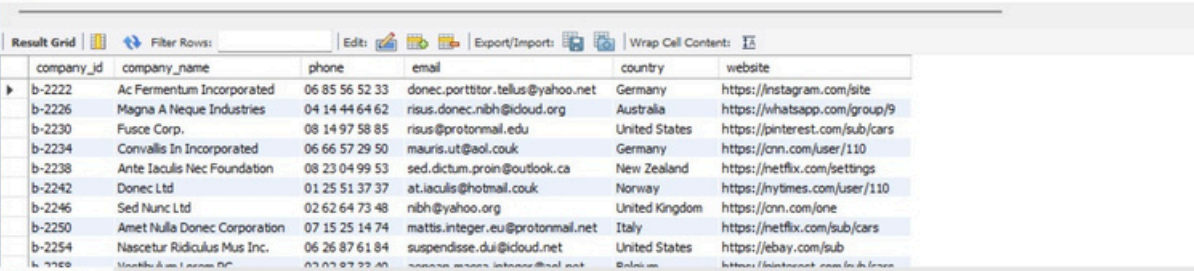
Action Output

#	Time	Action	Message
1	20:12:43	SELECT * FROM companies	0 row(s) returned

Después procederemos a la introducción de los datos:

Usamos Load data infile para la carga del archivo csv, se pone la ruta del archivo, especificamos que los campos estén separados por comas, y que tome los datos encerrados por "" como un campo, y que ignore la primera línea. Y finalmente, comprobamos nuevamente que hayan cargado todos los datos en la tabla.

```
79 • LOAD DATA
80 INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv'
81 INTO TABLE companies
82 FIELDS TERMINATED BY ','
83 ENCLOSED BY '"'
84 IGNORE 1 ROWS;
85
86 • SELECT * FROM companies; -- comprobamos que se hayan cargado en introducido los datos en la tabla companies
87
```



The screenshot shows the 'companies' table in SQL Server Enterprise Manager. The table now contains 49 rows of data. The columns are: company_id, company_name, phone, email, country, and website. The data is loaded from a CSV file.

```
companies 49 x
```

Output

Action Output

#	Time	Action	Message
1	20:17:17	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\companies.csv' INTO TABLE comp...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
2	20:17:17	SELECT * FROM companies	100 row(s) returned

- Tabla credit_cards

Comprobamos que no exista datos en la tabla credit_cards.

```
98 • SELECT * FROM credit_cards;
99
```

Result Grid									
Filter Rows:									
Edit: Export/Import: Wrap Cell Content:									
	id	user_id	iban	pan	pin	cvv	track1	track2	expiring_date
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

credit_cards 58 x credit_cards 59			
Output			
Action Output			
#	Time	Action	Message
✓ 1	20:55:35	SELECT * FROM credit_cards	0 row(s) returned

Cargamos los datos del archivo csv correspondiente a credit_cards e introducimos los datos, especificamos que separe los campos terminados en comas y que todo lo que este entre comillas lo considere un campo, e ignore la primera fila, al tener los encabezados ya creados en la creación de la tabla vacía. Verificamos nuevamente que hayan cargado todos los datos en la tabla.

```
88 -- tabla credit_cards
89 • SELECT * FROM credit_cards;
90
91 • LOAD DATA
92 INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\credit_cards.csv'
93 INTO TABLE credit_cards
94 FIELDS TERMINATED BY ','
95 ENCLOSED BY '"'
96 IGNORE 1 ROWS;
97
98 • SELECT * FROM credit_cards;
99
```

Result Grid									
Filter Rows:			Edit:		Export/Import:		Wrap Cell Contents:		
	id	user_id	iban	pan	pin	cvv	track1	track2	expiring_date
▶	CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%88383712448554646^WovsxegDpwiev^8604...	%87653863056044187=800716333673	10/30/22
	CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%84621311609958661^UfuyfsSeimxn^06106...	%84149568437843501=510714033071	08/24/23
	CcU-2952	273	BG451VQL52710525608255	4556 453 55 5287	4598	438	%82183285104307501^CddyttUxwfdq^5907...	%86778580257827162=6906859740077	06/29/21
	CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%87281111956795320^XocddjBkced^09016...	%84246154489281853=280522391678	02/24/23
	CcU-2966	271	BG72LKTQ15627628377363	448566 886747 7265	4900	130	%84728932322756223^HlgvufBmwig^7202...	%82318571115599881=890821578475	10/29/24
	CcU-2973	270	PT87806228135092429456346	544 58654 54343 384	8760	887	%84761405253275637^Hjnnip08lej1^7108515...	%87816169831446746=1310277229	01/30/25
	CcU-2980	269	DE39241881883086277136	402400 7145845969	5075	596	%87320483593870549^OakzqxHpsed^4901...	%82474313962214151=041221913175	07/24/22
	CcU-2987	268	GE89681434837748781813	3763 747687 76666	2298	797	%84750646345146674^PjmyrfGwvtrf^83051...	%85441935173418615=410370453677	10/31/23
	CcU-2994	267	BH62714428368066765294	344283273252593	7545	595	%81583759784015674^GmqoyhtUtoqrm^2507...	%84141467473024349=6506800955074	02/28/22
	CcU-2999	266	PL44074766474407476077110	511773 074073 7744	0673	067	%867770876770840^AulldfEmndu^7040	%87347075670840749=5706707073	04/16/23
credit_cards 58 x credit_cards 59									
Output									
Action Output									
#	Time	Action	Message						
✓ 1	20:55:35	SELECT * FROM credit_cards	0 row(s) returned						
✓ 2	20:55:35	LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\credit_cards.csv' INTO TABLE credit_cards	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0						
✓ 3	20:55:35	SELECT * FROM credit_cards	275 row(s) returned						

- Tabla products

Hacemos lo mismo que para los anteriores, verificamos que no tenga datos.

Cargamos el archivo csv de products, especificamos que este separado por comas cada campo, que incluya lo que está encerrado entre comillas como un campo, y que ignore la primera fila.

Verificamos nuevamente que hayan cargado todos los datos en la tabla.

```
100 -- tabla products
101 • SELECT * FROM products;
102
103 • LOAD DATA
104 INFIL 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv'
105 INTO TABLE products
106 FIELDS TERMINATED BY ','
107 ENCLOSED BY '"'
108 IGNORE 1 ROWS;
109
110 • SELECT * FROM products;
111
```

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	\$161.11	#7c7c7c	1	WH-4
2	Tarly Stark	\$9.24	#919191	2	WH-3
3	duel tourney Lannister	\$171.13	#d8d8d8	1.5	WH-2
4	warden south duel	\$71.89	#111111	3	WH-1
5	skywalker ewok	\$171.22	#bdbdbd	3.2	WH-0
6	dooku solo	\$136.60	#c4c4c4	0.8	WH-1
7	north of Casterly	\$63.33	#b7b7b7	0.6	WH-2
8	Winterfell	\$32.37	#383838	1.4	WH-3
9	Winterfell	\$76.40	#b5b5b5	1.2	WH-4

products 62 products 63 x

Output

Action Output

#	Time	Action	Message
✓ 5	21:10:39	SELECT * FROM products	0 row(s) returned
✓ 6	21:12:53	SELECT * FROM products	0 row(s) returned
✓ 7	21:12:53	LOAD DATA INFIL 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\products.csv' INTO TABLE produ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
✓ 8	21:12:53	SELECT * FROM products	100 row(s) returned

#	Time	Action	Message
✓	1	22:02:32 SELECT * FROM transactions	0 row(s) returned
✓	2	22:02:32 LOAD DATA INFILE 'C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\transactions.csv' INTO TABLE trans...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0
✓	3	22:02:32 SELECT * FROM transactions	587 row(s) returned

- Tabla users

Para montar esta tabla, hay 3 archivos csv que tienen la misma estructura de campos, es por eso por lo que adjuntamos una a una cada archivo a la tabla para formar una sola llamada users.

Los datos están separados por comas, y en este caso especificamos:

“ LINES TERMINATED BY '\r\n' ”, este permite leer archivos csv que provienen de Windows, solo hemos usado este formato para estos archivos de users, en el resto no ha hecho falta.

```
125 -- tabla users
126 • SELECT * FROM users;
127
128 • LOAD DATA
129 INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_uk.csv'
130 INTO TABLE users
131 FIELDS TERMINATED BY ','
132 ENCLOSED BY '"'
133 LINES TERMINATED BY '\r\n'
134 IGNORE 1 ROWS;
135
136 • LOAD DATA
```

The screenshot shows the MySQL Workbench interface. At the top, the SQL editor contains a script to create the 'users' table and load data from a CSV file. Below the editor, the 'Result Grid' shows the table structure with columns: id, name, surname, phone, email, birth_date, country, city, postal_code, and address. The 'Output' pane at the bottom shows the execution of the 'LOAD DATA' command, indicating that 50 rows were affected.

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.
152	Hakeem	Alford	(0111) 367 0184	adpiscung.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 Consectetur 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	B9H 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

Se comprueba que se hayan ingresado los datos:

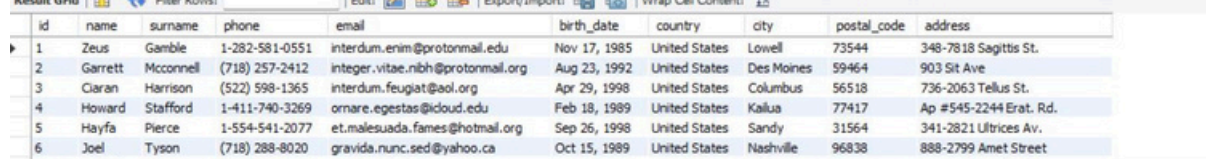
```
136 • SELECT * FROM users;
137
```

The screenshot shows the MySQL Workbench interface. The 'Result Grid' now displays the data loaded from the CSV file. The 'Output' pane shows the execution of the 'SELECT * FROM users;' command, indicating that 50 rows were returned.

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.
152	Hakeem	Alford	(0111) 367 0184	adpiscung.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 Consectetur 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	B9H 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

Luego continuamos ingresando los datos de los archivos csv restantes:

```
137
138 • LOAD DATA
139   INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_ca.csv'
140   INTO TABLE users
141   FIELDS TERMINATED BY ','
142   ENCLOSED BY '"'
143   LINES TERMINATED BY '\\r\\n'
144   IGNORE 1 ROWS;
145
146 • LOAD DATA
147   INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_usa.csv'
148   INTO TABLE users
149   FIELDS TERMINATED BY ','
150   ENCLOSED BY '"'
151   LINES TERMINATED BY '\\r\\n'
152   IGNORE 1 ROWS;
153
154 • SELECT * FROM users;
155
```



#	id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544	348-7818 Sagittis St.
2	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	903 Sit Ave
3	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbus	56518	736-2063 Tellus St.
4	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	77417	Ap #545-2244 Erat. Rd.
5	5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Sandy	31564	341-2821 Ultrices Av.
6	6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	Nashville	96838	888-2799 Amet Street

users 4 x

Output

Action Output

#	Time	Action	Message
2	10:37:07	LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\users_usa.csv' INTO TABLE users...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0
3	10:37:07	SELECT * FROM users	275 row(s) returned

CREAR CONSTRAINT

Para relacionar las tablas crearemos las constraint de la foreign key, relacionaremos las tablas users, companies y credit_cards con la tabla transactions.

```
157 -- se crean los constraint para relacionar las tablas
158 • ALTER TABLE transactions
159   ADD CONSTRAINT fk_trans_users
160   FOREIGN KEY(user_id) REFERENCES users(id);
161
162 • ALTER TABLE transactions
163   ADD CONSTRAINT fk_trans_comp
164   FOREIGN KEY(business_id) REFERENCES companies(company_id);
165
166 • ALTER TABLE transactions
167   ADD CONSTRAINT fk_trans_credit
168   FOREIGN KEY(card_id) REFERENCES credit_cards(id);
169
```

Output

Action Output

#	Time	Action	Message
1	10:40:36	ALTER TABLE transactions ADD CONSTRAINT fk_trans_users FOREIGN KEY(user_id) REFERENCES users(id)	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
2	10:40:36	ALTER TABLE transactions ADD CONSTRAINT fk_trans_comp FOREIGN KEY(business_id) REFERENCES companie...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0
3	10:40:37	ALTER TABLE transactions ADD CONSTRAINT fk_trans_credit FOREIGN KEY(card_id) REFERENCES credit_cards(id)	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0

Para relacionar la tabla productos con transactions, nos da un error:

'Error Code: 3780.Referencingcolumn 'product_ids'andreferencedcolumn 'id' in foreign key constraint'fk_trans_prod'are incompatible'.

Este error se refiere a que las columnas que queremos relacionar en ambas tablas son incompatibles y se refiere a que en la tabla transactions la columna product_ids contiene varios valores en cada celda, que son los productos comprados en esa transaccion.

The screenshot shows a database query interface. At the top, a SQL query is entered:

```
174 • SELECT product_ids
175 FROM TRANSACTIONS;
176
177
178 • SELECT n id
```

Below the query, the 'Result Grid' is displayed. It has a header 'product_ids' and a list of values, each on a new row:

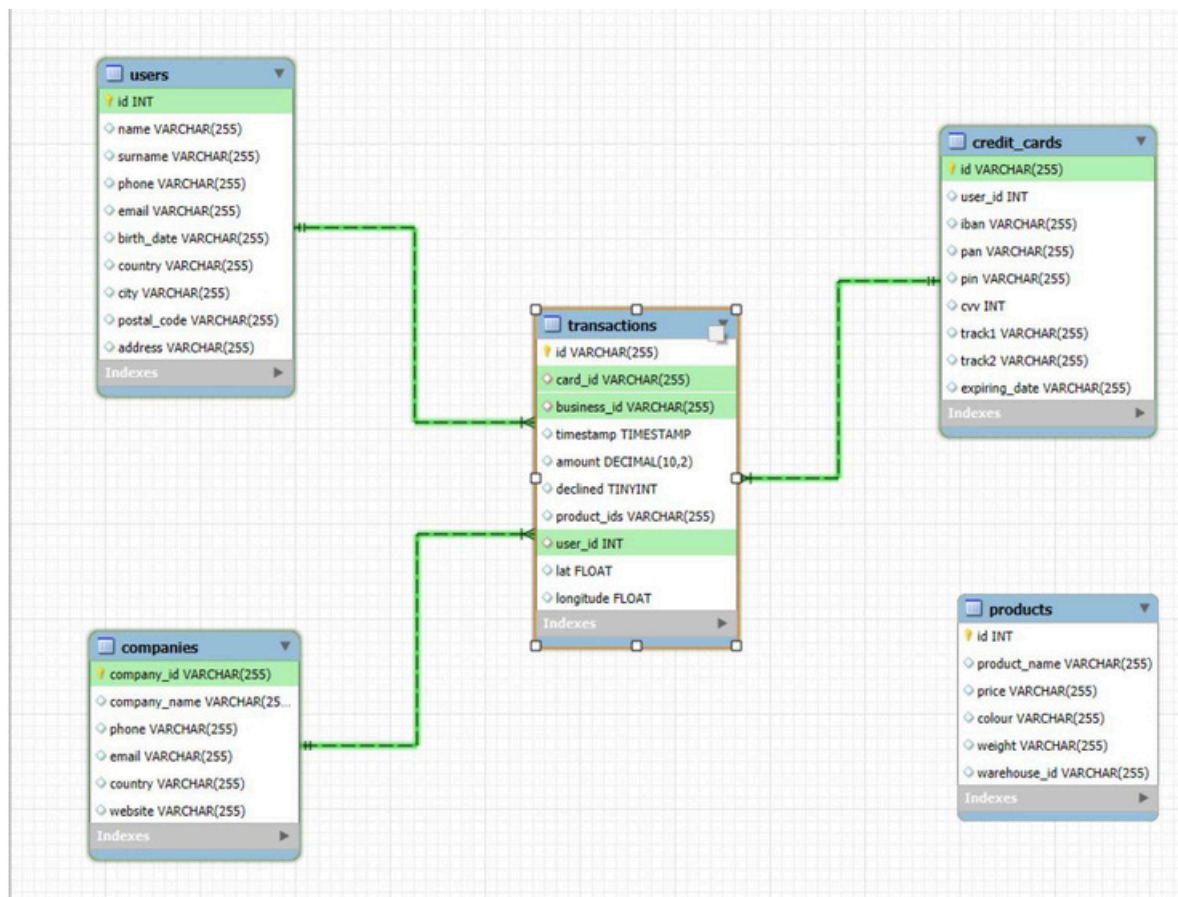
- 71, 1, 19
- 47, 97, 43
- 47, 67, 31, 5
- 89, 83, 79
- 43, 31
- 47, 23
- 67, 7
- 29, 41, 11
- 19, 41, 29, 3
- 89, 31
- 83, 43, 73, 61
- 7, 47, 17
- 27, 12

Below the result grid, there is a tab labeled 'TRANSACTIONS 5'. At the bottom, the 'Output' section shows the execution of the query:

#	Time	Action	Message
1	11:02:27	SELECT product_ids FROM TRANSACTIONS	587 row(s) returned

Para nuestro proyecto, dejaremos momentáneamente sin relacionar la tabla products, y quedaría nuestro diagrama de la siguiente forma:

Sería un modelo en estrella donde la tabla de hechos es transactions y las tablas dimensiones serían: users, credit_cards, companies.



- Exercici 1

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

```

178 -- - Ejercicio 1
179 -- Realiza una subconsulta que muestre a todos los usuarios con más de 30 transacciones utilizando al menos 2 tablas.
180
181 • SELECT t.user_id, COUNT(*) AS Num_Transacciones
182 FROM transactions t
183 WHERE t.user_id = ANY (SELECT u.id FROM users u)
184 GROUP BY t.user_id
185 HAVING Num_Transacciones > 30
186 ORDER BY Num_Transacciones DESC;
187
  
```

Result Grid		
	user_id	Num_Transacciones
▶	272	76
	267	52
	275	48
	92	39

Result 3 x

Output

Action Output

#	Time	Action	Message
✓ 1	21:05:56	SELECT t.user_id, COUNT(*) AS Num_Transacciones FROM transactions t WHERE t.user_id = ANY (SELECT u.id F...	4 row(s) returned

- Exercici 2

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

```
188 -- - Ejercicio 2
189 -- Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.
190
191 • SELECT round(AVG(t.amount),2) AS Media_Amount_Donec
192 FROM transactions t
193 INNER JOIN credit_cards cr ON t.card_id = cr.id
194 INNER JOIN companies c ON t.business_id = c.company_id
195 WHERE c.company_name = 'Donec Ltd';
196
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Media_Amount_Donec			
203.72			

Result 5 ×

Output

Action Output

#	Time	Action	Message
✓ 1	21:10:04	SELECT round(AVG(t.amount),2) AS Media_Amount_Donec FROM transactions t INNER JOIN credit_cards cr ON t.c...	1 row(s) returned

Nivell 2

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades i genera la següent consulta:

```
198 -- Nivell 2
199 -- Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat
200 -- en si les últimes tres transaccions van ser declinades i genera la següent consulta:
201 • CREATE TABLE activ_card AS
202     SELECT card_id,
203         CASE
204             WHEN sum(declined) >= 3 THEN 'no activo'
205             ELSE 'activo'
206         END AS actividad
207     FROM (SELECT card_id, declined,
208         ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_num
209         FROM transactions
210         ) AS hist_card
211     WHERE row_num <=3
212     GROUP BY card_id;
213
214 • SELECT * FROM activ_card;
```

The screenshot shows a database interface with a 'Result Grid' and an 'Output' window. The 'Result Grid' displays a list of credit card IDs and their status as 'activo'. The 'Output' window shows the execution of SQL commands and their results.

card_id	actividad
CcU-2938	activo
CcU-2945	activo
CcU-2952	activo
CcU-2959	activo
CcU-2966	activo
CcU-2973	activo
CcU-2980	activo
CcU-2987	activo
CcU-2994	activo
CcU-3001	activo
CcU-3008	activo
CcU-3015	activo

#	Time	Action	Message
1	12:08:30	CREATE TABLE activ_card AS SELECT card_id, CASE WHEN sum(declined) >= 3 THEN 'no activo' ELSE 'activo'	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0
2	12:08:30	SELECT * FROM activ_card	275 row(s) returned

Una vez creada esta tabla la relaciono con Credit_cards

```
215 -- Relaciono esta tabla con el modelo
216 • ALTER TABLE activ_card
217     ADD CONSTRAINT fk_credit FOREIGN KEY(card_id) REFERENCES credit_cards(id);
218
```

The screenshot shows the 'Output' window of a database interface, displaying the execution of SQL commands and their results.

#	Time	Action	Message
14	11:21:54	SELECT * FROM transacciones.credit_cards	275 row(s) returned
15	11:22:34	SELECT * FROM activ_card	275 row(s) returned
16	11:26:42	ALTER TABLE activ_card ADD CONSTRAINT fk_credit FOREIGN KEY(card_id) REFERENCES credit_cards(id)	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

- Exercici 1

Quantes targetes estan actives?

```
219  -- - Exercici 1
220  -- Quantes targetes estan actives?
221  • SELECT count(*)
222     FROM activ_card
223     WHERE actividad = 'activo';
```

Result Grid

count(*)
275

Result 44 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:10:28	SELECT count(*) FROM activ_card WHERE actividad = 'activo'	1 row(s) returned

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: Se ha separado la columna `products_ids` de la tabla `transactions`, ya que en cada celda había los id de productos vendidos en una transacción separados por comas, entonces se han separado con `Substring_index`, respetando la transacción a la que pertenece, con esto se han creado una tabla llamada `trans_product`.

```

226 • CREATE TABLE trans_product AS
227 SELECT id,
228     SUBSTRING_INDEX(product_ids, ',', 1) AS producto
229 FROM transactions
230 WHERE product_ids LIKE '%,'
231 UNION ALL
232     SELECT id,
233     SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', -2), ',', -1) AS producto
234 FROM transactions
235 WHERE product_ids LIKE '%,%,'
236 UNION ALL
237     SELECT id,
238     SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', 2), ',', -1) AS producto
239 FROM transactions
240 WHERE product_ids LIKE '%,%,%,'
241 UNION ALL
242     SELECT id,
243     SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', -2), ',', 1) AS producto
244 FROM transactions
245 WHERE product_ids LIKE '%,%,%,%,';
246
247 • SELECT *
248 FROM trans_product;

```

Result Grid

Filter Rows:

Exports:

Wrap Cell Contents:

Fetch rows:

	id	producto
▶	02C6201E-D90A-1859-B4EE-88D2986D3B02	71
	0466A42E-47CF-8D24-FD01-C0B689713128	47
	063FBA79-99EC-66FB-29F7-25726D1764A5	47
	0668296C-CD89-A883-76BC-2E4C4F8CBAE	89
	06CD9AA5-9B42-D684-DDDD-A5E394FEBAA9	43
	07A46D4B-31A3-7E87-65B9-0DA902AD109F	47
	09DE92CE-6F27-2B87-1385-9385B2B388E2	67

trans_product 91 x

Output

Action Output

#	Time	Action	Message
1	20:31:17	CREATE TABLE trans_product AS SELECT id, SUBSTRING_INDEX(product_ids, ',', 1) AS producto FROM transactions	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
2	20:31:17	SELECT * FROM trans_product	1457 row(s) returned

Con esta nueva tabla creada, la relacionaremos con transactions y con products, para establecer una entidad – relación y que sirva de nexo o tabla puente para relacionarlas.

Así que modificamos el tipo de dato en la columna producto de la tabla trans_producto INT, para que se iguale en el tipo a la columna id de la tabla products.

También creamos una primary key compuesta en la tabla trans_product con los dos campos que tiene esta tabla (id y producto)

```

255
256 • ALTER TABLE trans_product ADD CONSTRAINT primary key (id,producto);
257

```

Output			
Action Output			
#	Time	Action	Message
1	11:31:38	ALTER TABLE trans_product ADD CONSTRAINT primary key (id,producto)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

Y finalmente, para relacionar Trans_product con trasactions y con products creamos las constraint de foreign keys.

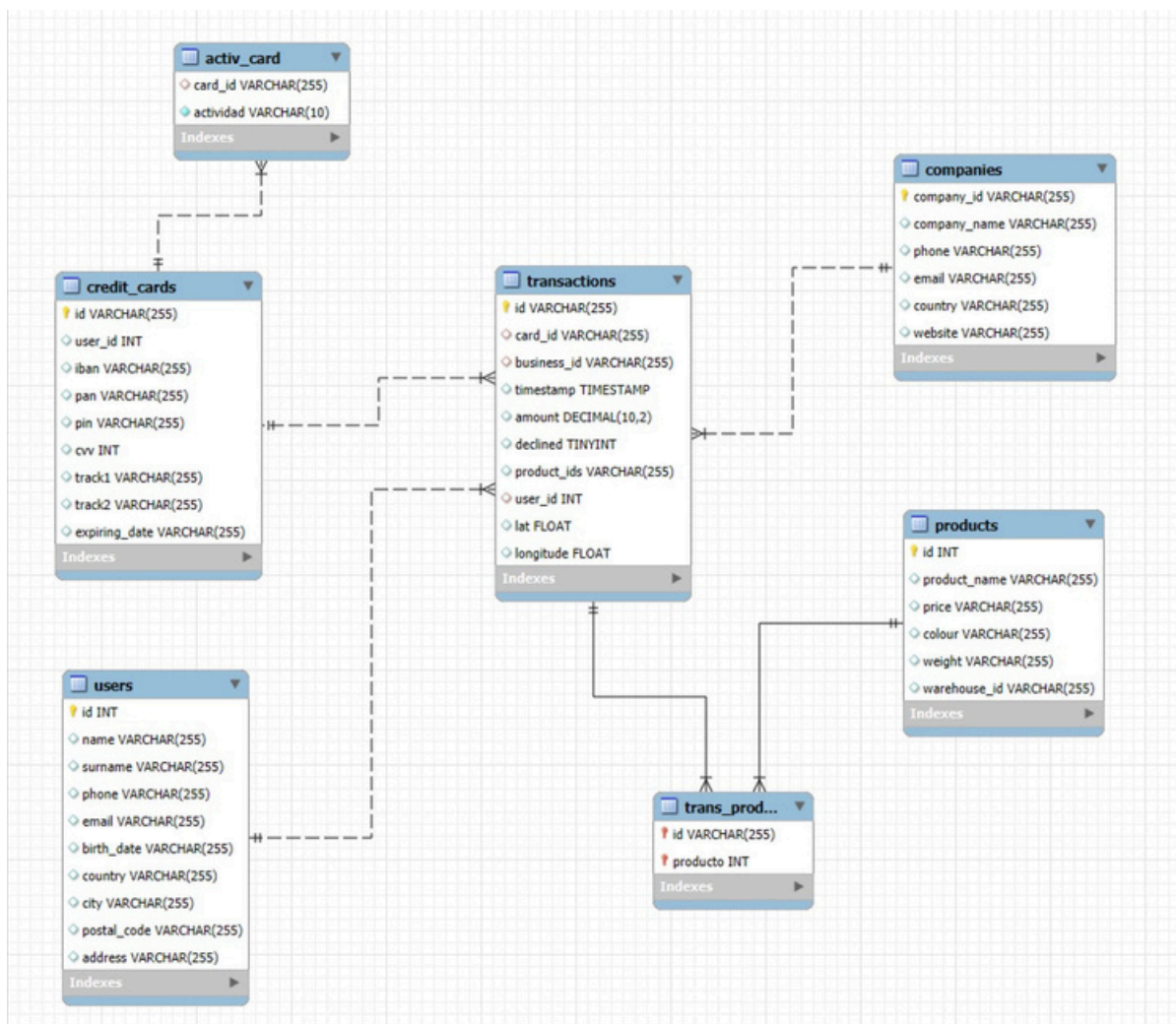
```

253 • ALTER TABLE trans_product MODIFY producto INT; -- modificar tipo de dato a INT
254
255 • ALTER TABLE trans_product
256 ADD CONSTRAINT fk_product FOREIGN KEY(producto) REFERENCES products(id);
257
258 • ALTER TABLE trans_product
259 ADD CONSTRAINT fk_id FOREIGN KEY (id) REFERENCES transactions(id);
260
261

```

#	Time	Action	Message
1	22:42:35	ALTER TABLE trans_product MODIFY producto INT	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
2	22:42:35	ALTER TABLE trans_product ADD CONSTRAINT fk_product FOREIGN KEY(producto) REFERENCES products(id)	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
3	22:42:36	ALTER TABLE trans_product ADD CONSTRAINT fk_id FOREIGN KEY (id) REFERENCES transactions(id)	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Aquí muestro el diagrama final y sus relaciones.





Ejercicio 1

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

Aquí mostramos cuantas veces se ha vendido cada producto, Mostramos un listado de productos con la cantidad de ventas de cada producto que se ha vendido, excluimos las ventas hechas y que hayan sido rechazadas.


```
269 • SELECT tp.producto as product, COUNT(tp.id) AS Veces_vendido
270 FROM trans_product tp
271 INNER JOIN transactions t ON tp.id = t.id
272 WHERE t.declined = '0'
273 GROUP BY product
274 ORDER BY product;
275
```

Result Grid   Filter Rows: Export:  Wrap Cell Content: 

	product	Veces_vendido
▶	1	51
	2	56
	3	43
	5	42
	7	44

Result 5 ×

Output

 Action Output

#	Time	Action	Message
✓ 1	18:27:32	SELECT tp.producto as product, COUNT(tp.id) AS Veces_vendido FROM trans_product tp INNER JOIN transactions ...	26 row(s) returned