

Sprint4. Sara Gutierrez Amigo

Descàrrega els arxius CSV, estudia'ls i dissenya una base de dades amb un esquema d'estrella que contingui, almenys 4 taules de les quals puguem realitzar les següents consultes:

Crearem primer la base de dades:

```
1 • CREATE DATABASE Sprint4;
```

✓	36	13:27:56	CREATE DATABASE Sprint4	1 row(s) affected	0.000 sec
---	----	----------	-------------------------	-------------------	-----------

Le decimos que queremos trabajar con esta base de datos y empezamos a crear las tablas:

```
3 • USE Sprint4;
4 • CREATE TABLE products ( id INT PRIMARY KEY,
5   Product_name VARCHAR (100),
6   Price VARCHAR (100),
7   Colour VARCHAR(100),
8   Weight DECIMAL (5,2),
9   Warehouse_id VARCHAR (100)
10 );
```

Una vez creada podemos empezar a importar los datos que tenemos de los archivos CSV con la siguiente query:

```
14 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'
15 INTO TABLE products
16 FIELDS TERMINATED BY ','
17 ENCLOSED BY '"'
18 LINES TERMINATED BY '\n'
19 IGNORE 1 ROWS;
```

Y comprobamos que todo está correcto:

```
SELECT * FROM products;
```

Sprint4. Sara Gutierrez Amigo

Result Grid

Edit:

Export/Imp

	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

✓

46 12:51:48 SELECT * FROM products

100 row(s) returned

0.000 sec / 0.000 sec

**ERRORES que nos hemos encontrado durante todo el proceso (aclaración antes de seguir con el resto de las tablas):*

1. La ubicación donde teníamos el archivo CSV no era la correcta y MySQL no lo encontraba. Para resolverlo teníamos que ir a buscar la ubicación desde donde MySQL coge los archivos y mover todos los CSV a esta ubicación. El comando que nos indica donde tenemos que ubicarlos es el siguiente:

```
SHOW VARIABLES LIKE 'secure_file_priv';
```

Aquí es donde tenemos que llevar todos los archivos:

Variable_name	Value
secure_file_priv	C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\

2. Equivocación en indicar el tipo de dato: por ejemplo, poner INT en datos numéricos donde necesitamos que nos tenga en cuenta los ceros iniciales (en ese caso no los contabilizará) o poner una cantidad de caracteres inferior, por ejemplo un VARCHAR muy pequeño. Al final casi todos los datos están puestos como VARCHAR (100) para evitar este tipo de errores. Menos las columnas TIMESTAMP que es mejor que vayan con formato DATETIME o las de longitud y latitud con FLOAT etc...

Las modificaciones las hemos ido haciendo con la siguiente query:

```
ALTER TABLE users_usa  
MODIFY COLUMN id VARCHAR(100);
```

Sprint4. Sara Gutierrez Amigo

3. Error a la hora de cargar los datos con el comando de `LOAD DATA INFILE` por equivocarnos en indicar cuales eran las separaciones entre columnas y al saltar de fila (`FIELDS TERMINATED BY` y `LINES TERMINATED BY`). Algunos archivos iban separados por coma, otros por punto y coma o terminaban en salto de línea y otros no ("`/n`" o "`/r/n`").

Vamos creando las otras tablas con el mismo proceso y vamos haciendo las comprobaciones:

```
25 • CREATE TABLE credit_cards (id VARCHAR (100) PRIMARY KEY,
26     user_id VARCHAR (100),
27     iban VARCHAR (100),
28     pan VARCHAR (100),
29     pin VARCHAR (100),
30     cvv VARCHAR (100),
31     track1 VARCHAR(150),
32     track2 VARCHAR(150),
33     expiring_date VARCHAR (100));
34
35 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
36 INTO TABLE credit_cards
37 FIELDS TERMINATED BY ','
38 ENCLOSED BY '"'
39 LINES TERMINATED BY '\n'
40 IGNORE 1 ROWS;
41
42 • SELECT * FROM credit_cards;
```

Result Grid									
Filter Rows:									
Edit: [Icons]									
Export/Import: [Icons]									
Wrap Cell Content: [Icon]									
	id	user_id	iban	pan	pin	cvv	track1	track2	
▶	CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%B8383712448554646^WovsxjDpwiev^8604...	%B7653863056044187=	
	CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%B4621311609958661^UftuyfsSeimxn^06106...	%B4149568437843501=	
	CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%B2183285104307501^CddytytUxwfdq^5907...	%B6778580257827162=	
	CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%B7281111956795320^XocddjBkecd^09016...	%B4246154489281853=	
	CcU-2966	271	BG72LKTQ15627628377363	448566 886747 7265	4900	130	%B4728932322756223^JhlgvsuFbmwgj^7202...	%B2318571115599881=	
	CcU-2973	270	PT87806228135092429456346	544 58654 54343 384	8760	887	%B4761405253275637^HjnnipoBlejrl^7108515...	%B7816169831446746=	
	CcU-2980	269	DE39241881883086277136	402400 7145845969	5075	596	%B7320483593870549^OokzqxHpsed^4901...	%B2474313962214151=	
	CcU-2987	268	GE89681434837748781813	3763 747687 76666	2298	797	%B4750646345146674^PjmlrfgWwvtrf^83051...	%B5441935173418615=	
	CcU-2994	267	BH62714428368066765294	344283273252593	7545	595	%B1583759784015674^GmqoyhtUtoqrn^2507...	%B4141467473024349=	
	CcU-3001	266	CY49087426654774581266832110	511722 924833 2244	9562	867	%B6227288756728648^AwxlfcdFmgvdy^2808...	%B3429355750963453=	
	CcU-3008	265	LU507216693616119230	4485744464433884	1856	740	%B7182449430529226^MlkoutyhTfdvpo^1708...	%B6235123731781366=	
	CcU-3015	264	PS119398216295715968342456821	3784 662733 17389	3246	822	%B5776250106724742^OuvzkrCwrvnm^530...	%B3561372148267521=	

✓ 107 14:42:39 SELECT * FROM credit_cards

275 row(s) returned

Sprint4. Sara Gutierrez Amigo

```
44 • CREATE TABLE companies (company_id VARCHAR (100) PRIMARY KEY,  
45     company_name VARCHAR (100),  
46     phone VARCHAR (100),  
47     email VARCHAR (100),  
48     country VARCHAR (100),  
49     website VARCHAR (100));  
50  
51 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'  
52 INTO TABLE companies  
53 FIELDS TERMINATED BY ','  
54 ENCLOSED BY ''''  
55 LINES TERMINATED BY '\n'  
56 IGNORE 1 ROWS;  
57  
58 • SELECT * FROM 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
	b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://netflix.com/sub/cars
	b-2254	Nascetur Ridiculus Mus Inc.	06 26 87 61 84	suspendisse.dui@icloud.net	United States	https://ebay.com/sub
	b-2258	Vestibulum Lorem PC	02 02 87 33 40	aenean.massa.integer@aol.net	Belgium	https://pinterest.com/sub/cars
	b-2262	Gravida Sagittis LLP	03 81 28 33 97	turpis.vitae@google.ca	Sweden	https://naver.com/site
	b-2266	Mus Aenean Eget Foundation	06 25 15 52 43	mi.duis@hotmail.net	Sweden	https://instagram.com/group/9
	b-2270	Dis Parturient Institute	05 36 29 78 74	purus@protonmail.org	Ireland	https://google.com/one
✓	108	14:44:18	SELECT * FROM companies			100 row(s) returned

Ahora las tablas de usuario:

Sprint4. Sara Gutierrez Amigo

```
60 • CREATE TABLE users_ca ( id VARCHAR (100) PRIMARY KEY,  
61   name VARCHAR (100),  
62   surname VARCHAR (100),  
63   phone VARCHAR (100),  
64   email VARCHAR (100),  
65   birth_date VARCHAR (100),  
66   country VARCHAR (100),  
67   city VARCHAR (100),  
68   postal_code VARCHAR (100),  
69   address VARCHAR (100));  
70  
71 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv'  
72 INTO TABLE users_ca  
73 FIELDS TERMINATED BY ','  
74 ENCLOSED BY ''''  
75 LINES TERMINATED BY '\r\n'  
76 IGNORE 1 ROWS;  
77  
78 • SELECT * FROM users_ca;
```

Result Grid		Filter Rows:			Edit:	Export/Import:		Wrap Cell Content:	
	id	name	surname	phone	email	birth_date	country	city	
▶	201	Iola	Powers	018-139-4717	ante.blandit@outlook.edu	Mar 20, 2000	Canada	Rigolet	
	202	Maxwell	Holden	045-402-7693	donec@hotmail.edu	Dec 2, 1986	Canada	Murdochville	
	203	Jarrold	Fields	010-741-8105	sit.amet@google.couk	Jan 6, 1982	Canada	Baddeck	
	204	Emerson	Sharp	068-138-9383	ante.iaculis@outlook.ca	Oct 15, 1994	Canada	Maple Creek	
	205	Sonya	Mckee	041-151-9737	magna.phasellus.dolor@google.ca	May 7, 1983	Canada	Dieppe	
	206	Harper	Hart	030-656-1670	fringilla.donec@outlook.net	Nov 17, 2000	Canada	QuÃbec City	
	207	Yvonne	Hatfield	003-854-1445	magna.et.ipsuam@google.edu	Sep 22, 1981	Canada	Rae-Edzo	
	208	Burke	Graham	064-568-4454	vel@yahoo.org	Feb 23, 1993	Canada	Annapolis Royal	
	209	Athena	Malone	027-280-8275	pellentesque.tincidunt@yahoo.ca	Dec 14, 1991	Canada	Cambridge Bay	
	210	Slade	Poole	084-771-1363	amet@icloud.com	Feb 16, 2001	Canada	Ottawa	

27 10:59:53 SELECT * FROM users_ca 75 row(s) returned 0.000 sec / 0.000 sec

Sprint4. Sara Gutierrez Amigo

```
80 • CREATE TABLE users_uk ( id VARCHAR (100) PRIMARY KEY,  
81     name VARCHAR (100),  
82     surname VARCHAR (100),  
83     phone VARCHAR (100),  
84     email VARCHAR (100),  
85     birth_date VARCHAR (100),  
86     country VARCHAR (100),  
87     city VARCHAR (100),  
88     postal_code VARCHAR (100),  
89     address VARCHAR (100));  
90  
91 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'  
92 INTO TABLE users_uk  
93 FIELDS TERMINATED BY ','  
94 ENCLOSED BY ''''  
95 LINES TERMINATED BY '\r\n'  
96 IGNORE 1 ROWS;  
97  
98 • SELECT * FROM users_uk;
```

Result Grid		Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:	
	id	name	surname	phone	email	birth_date	country	city	
▶	151	Meghan	Hayden	0800 746 6747	arcu.vel@hotmail.ca	Jul 2, 1980	United Kingdom	Tullit	
	152	Hakeem	Alford	(0111) 367 0184	adipiscing.ligula@google.edu	Sep 30, 1979	United Kingdom	Kett	
	153	Keegan	Pugh	(016977) 3851	sodales.nisi@aol.org	Jul 27, 1994	United Kingdom	Whit	
	154	Cooper	Bullock	(021) 2521 6627	et@outlook.net	Nov 2, 1986	United Kingdom	Pres	
	155	Joshua	Russell	055 4409 5286	justo.nec.ante@outlook.edu	Jan 23, 1984	United Kingdom	Hatf	
	156	Remedios	Case	055 3114 1566	mollis.phasellus.libero@aol.com	Oct 9, 1994	United Kingdom	Nort	
	157	Philip	Carey	0800 640 6251	phasellus@yahoo.net	Oct 10, 1992	United Kingdom	Loch	

✓ 19 10:56:05 SELECT * FROM users_uk 50 row(s) returned 0.000 sec / 0.000 sec

Sprint4. Sara Gutierrez Amigo

```
100 • CREATE TABLE users_usa ( id VARCHAR (100) PRIMARY KEY,  
101     name VARCHAR (100),  
102     surname VARCHAR (100),  
103     phone VARCHAR (100),  
104     email VARCHAR (100),  
105     birth_date VARCHAR (100),  
106     country VARCHAR (100),  
107     city VARCHAR (100),  
108     postal_code VARCHAR (100),  
109     address VARCHAR (100));  
110  
111 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv'  
112 INTO TABLE users_usa  
113 FIELDS TERMINATED BY ','  
114 ENCLOSED BY ''''  
115 LINES TERMINATED BY '\r\n'  
116 IGNORE 1 ROWS;  
117  
118 • SELECT * FROM users_usa;
```

Result Grid		Filter Rows:			Edit:	Export/Import:		Wrap Cell Content:	
	id	name	surname	phone	email	birth_date	country	city	
▶	1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	
	2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moi	
	3	Ciaran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	Apr 29, 1998	United States	Columbi	
	4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	
	5	Hayfa	Pierce	1-554-541-2077	et.malesuada.fames@hotmail.org	Sep 26, 1998	United States	Sandy	
	6	Joel	Tyson	(718) 288-8020	gravida.nunc.sed@yahoo.ca	Oct 15, 1989	United States	Nashvill	
	7	Rafael	Jimenez	(817) 689-0478	eget@outlook.ca	Dec 4, 1981	United States	Hillsborc	

✓ 11 10:49:51 SELECT * FROM users_usa 150 row(s) returned 0.000 sec / 0.000 sec

Nos damos cuenta que tenemos que unificar las tres tablas de User porque si no, al crear la tabla de transacciones, la Foreign Key de User_id se va a tener que referenciar a 3 tablas distintas. Primero creamos una tabla nueva que va a tener los mismos campos que las tres tablas de usuarios anteriores más uno nuevo que le llamaremos “region” donde añadiremos los valores de UK, USA o CA:

Sprint4. Sara Gutierrez Amigo

```
122 • CREATE TABLE users ( id VARCHAR (100) PRIMARY KEY,  
123     name VARCHAR (100),  
124     surname VARCHAR (100),  
125     phone VARCHAR (100),  
126     email VARCHAR (100),  
127     birth_date VARCHAR (100),  
128     country VARCHAR (100),  
129     city VARCHAR (100),  
130     postal_code VARCHAR (100),  
131     address VARCHAR (100),  
132     region VARCHAR (10));
```





✓ 50 19:19:23 CREATE TABLE users (id INT PRIMARY KEY... 0 row(s) affected 0.016 sec

Luego introduciremos los datos de cada una de las tablas que ya están cargadas en la base de datos con el comando INSERT INTO, asignándoles en valor en “region” según de donde sea la región de origen: UK, USA o CA:

```
134 • INSERT INTO users (id, name, surname, phone, email, birth_date, country, city, postal_code, address, region)  
135 SELECT id, name, surname, phone, email, birth_date, country, city, postal_code, address, 'CA' FROM users_ca;  
136  
137 • INSERT INTO users (id, name, surname, phone, email, birth_date, country, city, postal_code, address, region)  
138 SELECT id, name, surname, phone, email, birth_date, country, city, postal_code, address, 'UK' FROM users_uk;  
139  
140 • INSERT INTO users (id, name, surname, phone, email, birth_date, country, city, postal_code, address, region)  
141 SELECT id, name, surname, phone, email, birth_date, country, city, postal_code, address, 'USA' FROM users_usa;
```

Y comprobamos:

```
143 • SELECT * FROM users;
```

Result Grid		 Filter Rows:	Edit:			
	id	name	email	region		
▶	1	Zeus	interdum.enim@protonmail.edu	USA		
	2	Garrett	integer.vitae.nibh@protonmail.org	USA		
	3	Ciaran	interdum.feugiat@aol.org	USA		
	4	Howard	ornare.egestas@icloud.edu	USA		
	5	Hayfa	et.malesuada.fames@hotmail.org	USA		
	6	Joel	gravida.nunc.sed@yahoo.ca	USA		

✓ 56 19:29:36 SELECT * FROM users 275 row(s) returned 0.000 sec / 0.000 sec

Vemos que las filas que nos devuelve son

275, la suma de las 3 tablas, por lo tanto, es correcto.

Sprint4. Sara Gutierrez Amigo

Y ahora borramos las tablas de Users que tenemos por separado, porque ya no las necesitamos:

99	•	DROP TABLE users_ca;		
100	•	DROP TABLE users_uk;		
101	•	DROP TABLE users_usa;		
✓	57	19:32:59	DROP TABLE users_ca	0 row(s) affected 0.016 sec
✓	58	19:33:03	DROP TABLE users_uk	0 row(s) affected 0.031 sec
✓	59	19:33:07	DROP TABLE users_usa	0 row(s) affected 0.031 sec

Por último, como ya hemos creado todas las tablas de dimensiones, ya podemos crear la tabla de hechos, que en nuestro caso es la de transactions:

```
147 • CREATE TABLE transactions ( id VARCHAR (100) PRIMARY KEY,
148     card_id VARCHAR (100),
149     business_id VARCHAR (100),
150     timestamp datetime,
151     amount DECIMAL (6,2),
152     decline INT,
153     product_ids VARCHAR (100),
154     user_id VARCHAR (100),
155     lat FLOAT,
156     longitude FLOAT,
157     FOREIGN KEY (card_id) REFERENCES credit_cards(id),
158     FOREIGN KEY (business_id) REFERENCES companies(company_id),
159     FOREIGN KEY (product_ids) REFERENCES products(id),
160     FOREIGN KEY (user_id) REFERENCES users(id));
```

Y cargamos los datos:

```
170 • LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
171 INTO TABLE transactions
172 FIELDS TERMINATED BY ';'
173 ENCLOSED BY '"'
174 LINES TERMINATED BY '\n'
175 IGNORE 1 ROWS;
```

Nos da el siguiente error: “Cannot add or update a child row: a foreign key constraint fails (`sprint4`.`transactions`, CONSTRAINT `transactions_ibfk_3` FOREIGN KEY (`product_ids`) REFERENCES `products` (`id`))”

Si miramos las tablas vemos que la Foreign Key que hemos creado para que se referencie con la Primary Key ID de la tabla de productos no se puede realizar, porque en la columna

Sprint4. Sara Gutierrez Amigo

de transacciones hay más de un valor. Es decir, para cada una de las transacciones realizadas se compra más de un producto. Por lo tanto la relación acaba siendo de muchos a muchos (N:N).

¿Como solucionamos este problema? Tenemos que crear una tabla intermedia donde se vea de manera desglosada cada una de las transacciones con solamente un producto. En esta tabla nueva, por tanto, cada transacción se va a repetir como productos se haya comprado dentro de ella.

Para hacer esto primero vamos a eliminar la Foreign Key que hemos creado en la tabla de transacciones. Como no le pusimos nombre, vamos a buscar primero como se llama con el comando de SHOW CREATE TABLE. Una vez que sabemos el nombre de nuestra clave foránea la eliminamos con el comando ALTER TABLE y DROP:

```
186 • SHOW CREATE TABLE transactions; -- Obtener el nombre de la clave foránea
187 • ALTER TABLE transactions DROP FOREIGN KEY transactions_ibfk_3;
```

Ahora creamos una nueva tabla donde vamos a migrar luego los datos del ID de transacciones y los productos comprados en cada una de ellas, solo necesitaremos por tanto dos campos:

```
192 • CREATE TABLE transaction_products (transaction_id VARCHAR(100),
193     product_id VARCHAR (100),
194     FOREIGN KEY (transaction_id) REFERENCES transactions(id),
195     FOREIGN KEY (product_id) REFERENCES products(id));
```

Luego, con INSERT INTO, sabiendo que los valores de la columna Products_id de la tabla transactions están separados por comas y un espacio ejecutamos la siguiente query para que estos valores con TRIM queden ya separados:

```
INSERT INTO transaction_products (transaction_id, product_id)
SELECT
    id AS transaction_id,
    TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(REPLACE(product_ids, ' ', ''), ' ', numbers.n), ' ', -1)) AS product_id
FROM
    (SELECT 1 AS n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5 UNION ALL SELECT 6 UNION ALL SELECT 7 UNION ALL SELECT 8 UNION ALL SELECT 9 UNION ALL SELECT 10) numbers
INNER JOIN transactions ON CHAR_LENGTH(REPLACE(product_ids, ' ', '')) - CHAR_LENGTH(REPLACE(REPLACE(product_ids, ' ', ''), ' ', '')) >= numbers.n - 1;
```

(Ampliado):

```
199 • INSERT INTO transaction_products (transaction_id, product_id)
200     SELECT
201         id AS transaction_id,
202         TRIM(SUBSTRING_INDEX(SUBSTRING_INDEX(REPLACE(product_ids, ' ', ''), ' ', numbers.n)
203     FROM
204     (SELECT 1 AS n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5
205     INNER JOIN transactions ON CHAR_LENGTH(REPLACE(product_ids, ' ', '')) - CHAR_LENGTH
```

Sprint4. Sara Gutierrez Amigo

Vemos que ya hay un valor por cada uno de los productos:

```
209 • SELECT * FROM transaction_products;
```

Result Grid	Filter Rows:	Export:
transaction_id	product_id	
FE96CE47-BD59-381C-4E18-E3CA3D44E8FF	3	
FE809ED4-2DB6-55AC-C915-929516E4646B	43	
FE809ED4-2DB6-55AC-C915-929516E4646B	23	
FD9CBCCD-8E1E-8DA1-4606-7E3A6F3A5A65	37	
FD89D51B-AE8D-77DC-E450-B8083FBD3187	1	
FD89D51B-AE8D-77DC-E450-B8083FBD3187	73	
FD89D51B-AE8D-77DC-E450-B8083FBD3187	2	
FD89D51B-AE8D-77DC-E450-B8083FBD3187	3	
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	89	
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	17	
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	7	
FD2E8957-414B-BEEC-E9AD-59AA7A8A6290	83	
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	37	
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	41	
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	5	
FCE2AB9A-271D-2BDC-9E49-8DD92A373391	59	
FBD7E0D6-8A6B-F5BC-0CA9-EA4B8760100C	29	

109 15:31:21 SELECT * FROM transaction_products 1457 row(s) returned 0.000 sec / 0.000 s

Comprobamos que todo esté correcto cogiendo un Id de transacción al azar:

```
216 • SELECT product_ids, id FROM transactions WHERE ID = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
```

Result Grid	Filter Rows:	Edit:
product_ids	id	
71, 1, 19	02C6201E-D90A-1859-B4EE-88D2986D3B02	
NULL	NULL	

Según esto en la tabla transaction_products nos tendría que salir lo siguiente:

Result Grid	Filter Rows:	Export:
transaction_id	product_id	
02C6201E-D90A-1859-B4EE-88D2986D3B02	19	
02C6201E-D90A-1859-B4EE-88D2986D3B02	1	
02C6201E-D90A-1859-B4EE-88D2986D3B02	71	

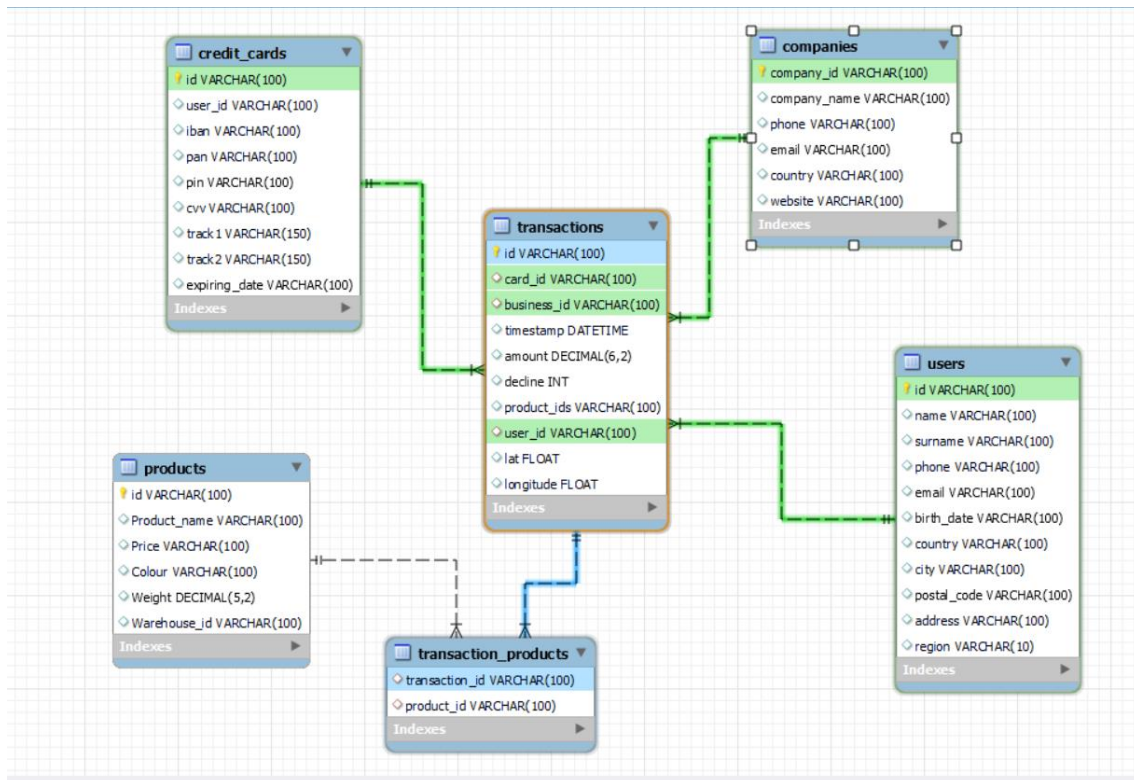
```
219 • SELECT * FROM transaction_products
220 where transaction_id = "02C6201E-D90A-1859-B4EE-88D2986D3B02";
```

Es correcto.

Sprint4. Sara Gutierrez Amigo

Finalmente, el modelo y relaciones entre tablas queda de la siguiente manera:

Un modelo estrella donde la tabla de hechos es la de transactions y está relacionada con las tablas de dimensiones de 1 a muchos.



- Exercici 1

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

O bien podemos hacerlo de esta manera:

```
226 • SELECT name, surname, users.id
227   from users
228   JOIN (SELECT count(id) as numtransactions, user_id
229         FROM transactions
230        GROUP BY user_id
231        having numtransactions > 30) as tabletransactions
232  ON users.id = tabletransactions.user_id;
```

Sprint4. Sara Gutierrez Amigo

Haremos una subquery utilizando la tabla de transactions. Esta tabla necesitamos que esté agrupada por el id de usuario y que cuente cuantas transacciones ha habido por usuario. Esta selección la pondremos bajo la condición de que ese conteo sea mayor a 30, que solo nos seleccione las que son mayores a 30 con el comando HAVING.

Esta tabla creada con esta subquery es la que vamos a unir con la tabla de usuarios para que la tabla users nos seleccione el nombre, apellidos y el número de usuario relacionándose a través de este número de usuario entre ellas dos.

	name	surname	id
▶	Ocean	Nelson	267
	Hedwig	Gilbert	272
	Kenyon	Hartman	275
	Lynn	Riddle	92

La otra manera en la que lo podemos hacer es esta:

```
237 • SELECT name, surname
238 FROM users
239 WHERE id IN (
240     SELECT user_id
241     FROM transactions
242     GROUP BY user_id
243     HAVING COUNT(id) > 30);
```

En este caso lo haremos directamente trabajamos con la tabla users. Seleccionamos el nombre y el apellido. Y del campo de ID diremos que nos seleccione lo que le hemos indicado dentro de la condición WHERE IN con una subquery.

En esta subquery utilizamos la tabla transactions: El campo con el que estamos trabajando y con el que podemos relacionar las dos tablas es el ID de usuario. Por tanto, en esta subquery vamos a seleccionar el Id de usuario de la tabla de transactions, agrupado por el usuario y poniendo la condición que el conteo del ID de transacciones por usuario sea mayor a 30. Esta query seleccionará solo los ID de usuario que cumplan esa condición.

Sprint4. Sara Gutierrez Amigo

	name	surname
▶	Ocean	Nelson
	Hedwig	Gilbert
	Kenyon	Hartman
	Lynn	Riddle

- Exercici 2

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

```
247 • SELECT round(avg(amount),2), iban, company_name
248 FROM transactions
249 JOIN credit_cards
250 ON transactions.card_id = credit_cards.id
251 JOIN companies
252 ON transactions.business_id = companies.company_id
253 WHERE company_name = "Donec Ltd"
254 GROUP BY iban;
```

En este caso uniremos tres tablas. Porque necesitamos la tabla de credit cards para utilizar el IBAN, la tabla de companies para poner la condición de que solo queremos la compañía llamada Donec Ltd y por último el campo de amount de la tabla de transacciones.

Seleccionaremos la media de amount (donde le hemos puesto la condición que nos redondee la media a máximo dos decimales con el comando ROUND), el IBAN y el nombre de la compañía. Después de ir uniendo las tablas mediante los JOIN relacionando las PK con las FK, pondremos la condición de que nos seleccione solo la compañía llamada Donec Ltd y que nos lo agrupe por IBAN.

	round(avg(amount),2)	iban	company_name
▶	203.72	PT87806228135092429456346	Donec Ltd

Si lo hubiéramos querido hacer solo por con dos tablas como dice el enunciado tendríamos que haber averiguado primero que Id de compañía corresponde a Donec Ltd en la tabla de companies porque a través de ahí pondríamos esa condición con la tabla de transacciones. Porque la relación entre ellas es con el campo de id de business.

Sprint4. Sara Gutierrez Amigo

Una vez que lo averiguamos uniríamos la tabla de credit cards con transacciones a través de un JOIN y las condiciones serían la agrupación por IBAN y que el business ID sea el que nos ha dado la query anterior, más la selección de la media del amount de transacciones:

```
26 • SELECT company_id
27 FROM companies
28 WHERE company_name = "Donec Ltd";
29
30 -- Sabemos ahora que el business_id de transactions tiene que ser = b-2242.
31
32 • SELECT avg(amount), iban, business_id
33 FROM transactions
34 JOIN credit_cards
35 ON transactions.card_id = credit_cards.id
36 WHERE business_id = "b-2242"
37 GROUP BY iban;
```

Result Grid	
	company_id
▶	b-2242
*	NULL

Result Grid		Filter Rows:	Export:
	avg(amount)	iban	business_id
▶	203.715000	PT87806228135092429456346	b-2242