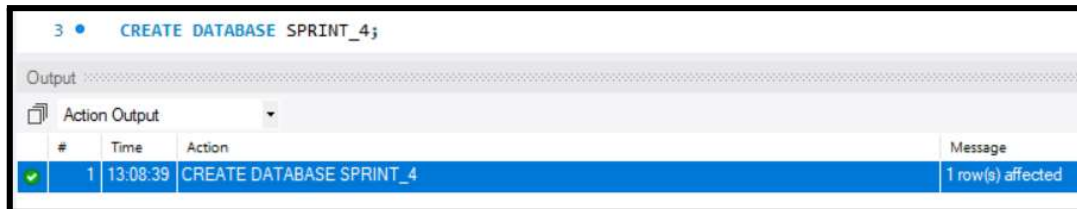


SPRINT 4

NIVELL 1

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

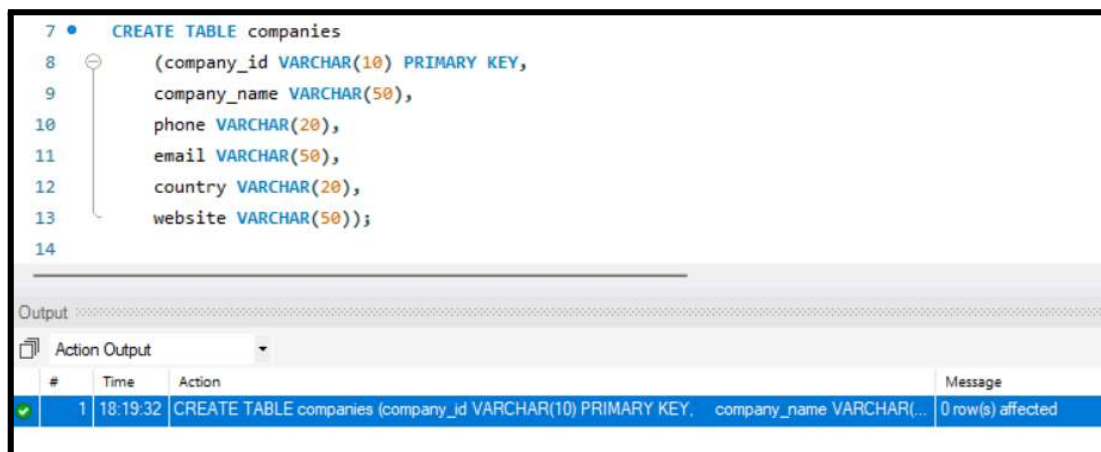
Primero creo una BBD nueva llamada Sprint_4:



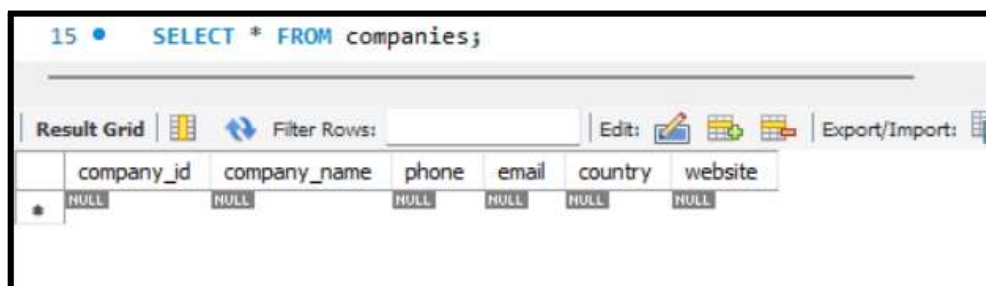
Comienzo con la creación de la estructura de las tablas y definición del formato según el contenido del fichero csv: VARCHAR para cadenas, INT para números enteros, etc.

TABLA de dimensiones "companies":

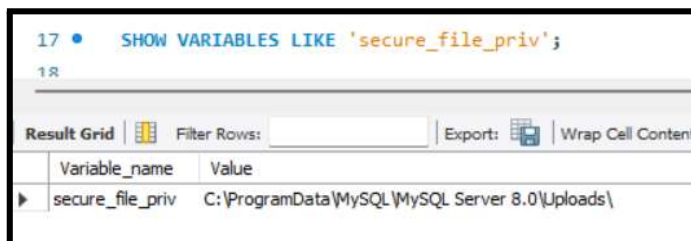
- PRIMARY KEY: campo id que contiene identificadores únicos para cada uno de los registros
- Resto de campos: atributos de las empresas.



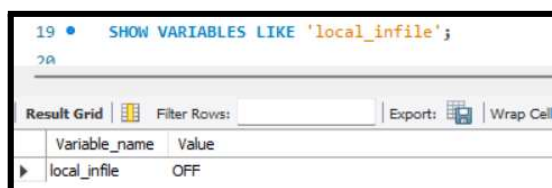
Compruebo que se haya creado la estructura correctamente mediante la cláusula SELECT:



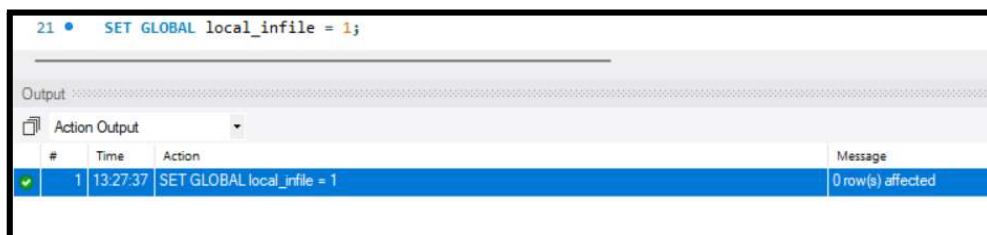
Antes de importar, es necesario saber el directorio que MySQL tiene permitido usar para la importación de datos externos:



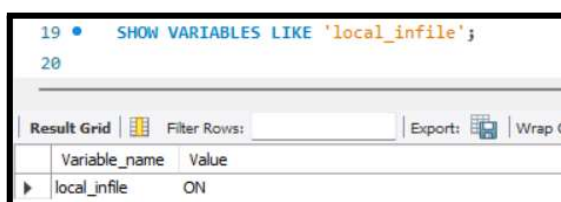
Verifico también el estado de la funcionalidad de carga de archivos locales:



Si el valor es “OFF” significa que está deshabilitada la función que permite la carga de archivos locales y, por tanto, hay que habilitarla mediante esta sentencia:



Comprobamos que ya aparece “ON” por lo que la función está habilitada:



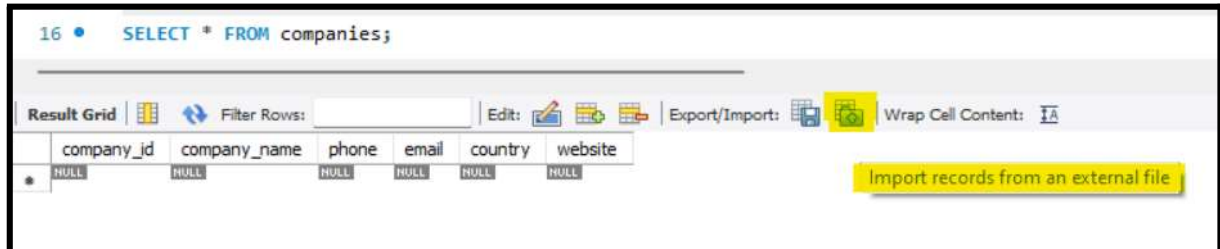
Este comando carga los datos del archivo “companies.csv” desde el directorio “C:\Program Files\MySQL\MySQL Server 8.0\Uploads\companies.csv”, que es el directorio permitido, y los inserta en la tabla companies. En la sentencia también indicamos las características del fichero:

- Los campos en el archivo CSV están separados por comas (línea de código 26)
- Están encerrados entre comillas dobles (27)
- Y además, la primera fila del archivo se ignora ya que contiene encabezados (28):



NOTA A LUCÍA: No he podido cargar los datos mediante código ya que me da un error de seguridad que no consigo corregir. Realizo la carga mediante la siguiente alternativa:

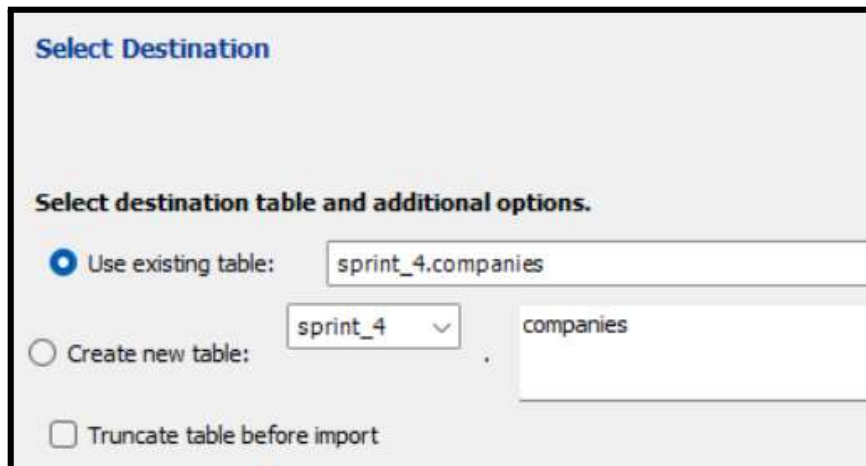
Visualizar la tabla y seleccionar la opción "Import record from an external file":



Seleccionar directorio:




Marcar la tabla ya existente "companies":



Comprobar que la correspondencia entre los campos de la tabla y las cabeceras del archivo sea correcta y que los valores de cada campo estén correctamente posicionados:


Configure Import Settings

Detected file format: csv 

Encoding: utf-8

Columns:

Source Column	Dest Column
<input checked="" type="checkbox"/> company_id	company_id
<input checked="" type="checkbox"/> company_name	company_nam
<input checked="" type="checkbox"/> phone	phone
<input checked="" type="checkbox"/> email	email
<input checked="" type="checkbox"/> country	country
<input checked="" type="checkbox"/> website	website



company_id	company_name	phone	email	country	website
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo...	Germany	https://instagram.co.
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.co.
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany	https://cnn.com/user

Para finalizar, comparar el mensaje con el fichero csv para comprobar que la importación se ha realizado con éxito (nota: en el fichero cvs salen 101 registros ya que incluye los encabezados):

Import Results

File C:\csv\companies.csv was imported in 0.338 s

Table sprint_4.companies has been used

100 records imported

	A	B	C	D	E	F	G	H
1	company_id,company_name,phone,email,country,website							
2	b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://instagram.com/		
3	b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/group/S		
4	b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/sub/cars		
5	b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany	https://cnn.com/user/110		
6	b-2238	Ante Iaculis Nec Foundation	08 23 04 99 53	sed.dictum.proin@outlook.ca	New Zealand	https://netflix.com/setti		
7	b-2242	Donec Ltd	01 25 51 37 37	at.iaculis@hotmail.couk	Norway	https://nytimes.com/user/110		
8	b-2246	Sed Nunc Ltd	02 62 64 73 48	nibh@yahoo.org	United Kingdom	https://cnn.com/one		
9	b-2250	Amet Nulla Donec Corporation	07 15 25 14 74	mattis.integer.eu@protonmail.net	Italy	https://netflix.com/sub/ca		

companies

Recuento: 101

100 %

Comprobar:

16 • `SELECT * FROM companies;`

	company_id	company_name	phone	email	country	website
▶	b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://instagram.com/site
	b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@idoud.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@idoud.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

companies 4 x

Output

Action Output

#	Time	Action	Message
9	11:31:50	DEALLOCATE PREPARE stmt	OK
10	11:37:29	SELECT * FROM companies	100 row(s) returned

Repetir el proceso y crear todas las tablas de dimensiones: credit_cards, users_ca, users_uk y users_usa.

Hay 3 tablas con misma estructura que contienen los mismos datos de diferentes usuarios, la única diferencia entre ellas es el país de residencia. Como solo podremos relacionar una de ellas con la tabla de hechos, creo una nueva tabla unificando los datos de todos los usuarios y la almaceno con el nombre "total_users". Uso la cláusula "UNION" para descartar los duplicados:

86 • `CREATE TABLE total_users AS`
 87 `SELECT * FROM users_ca`
 88 `UNION SELECT * FROM users_uk`
 89 `UNION SELECT * FROM users_usa;`

Output

Action Output

#	Time	Action	Message
1	20:55:04	CREATE TABLE total_users AS SELECT * FROM users_ca UNION SELECT * FROM users_uk UNION ...	275 row(s) affected Records

Modifico del campo "id" para definirlo como PRIMARY KEY, obligatorio para establecer relaciones con otras tablas:

91 • `ALTER TABLE total_users ADD PRIMARY KEY (id);`
 92

Output

Action Output

#	Time	Action	Message
1	20:55:04	CREATE TABLE total_users AS SELECT * FROM users_ca UNION SELECT * FROM users_uk UNION ...	275 row(s)
2	20:55:51	ALTER TABLE total_users ADD PRIMARY KEY (id)	0 row(s) aff

Por último, creo la tabla de hechos “transactions” que tiene un identificador único “id” como PK, claves foráneas para relacionarla con las tablas de dimensiones: “credit_card_id”, bussines_id y “user_id”. El resto de campos son métricas atribuibles a cada “id” de cada transacción como la fecha, el importe, latitud y longitud y si la operación fue aceptada o declinada:

```

94 • CREATE TABLE transactions
95     (id VARCHAR(255) PRIMARY KEY,
96     credit_card_id VARCHAR(15),
97     business_id VARCHAR(20),
98     timestamp VARCHAR(50),
99     amount DECIMAL(10,2),
100    declined TINYINT(1),
101    product_ids VARCHAR(50),
102    user_id INT,
103    lat FLOAT,
104    longitude FLOAT,
105    FOREIGN KEY (credit_card_id) REFERENCES credit_cards(id),
106    FOREIGN KEY (user_id) REFERENCES total_users(id),
107    FOREIGN KEY (business_id) REFERENCES companies(company_id));

```

Output

#	Time	Action	Message
1	20:55:04	CREATE TABLE total_users AS SELECT * FROM users_ca UNION SELECT * FROM users_uk UNION ...	275 row(s) affected
2	20:55:51	ALTER TABLE total_users ADD PRIMARY KEY (id)	0 row(s) affected
3	20:57:06	CREATE TABLE transactions (id VARCHAR(255) PRIMARY KEY, credit_card_id VARCHAR(15), busi...	0 row(s) affected

Importo los datos del fichero csv siguiendo el procedimiento explicado anteriormente y compruebo que se hayan insertado bien todos los registros:

```

109 • SELECT * FROM transactions;
110

```

Result Grid

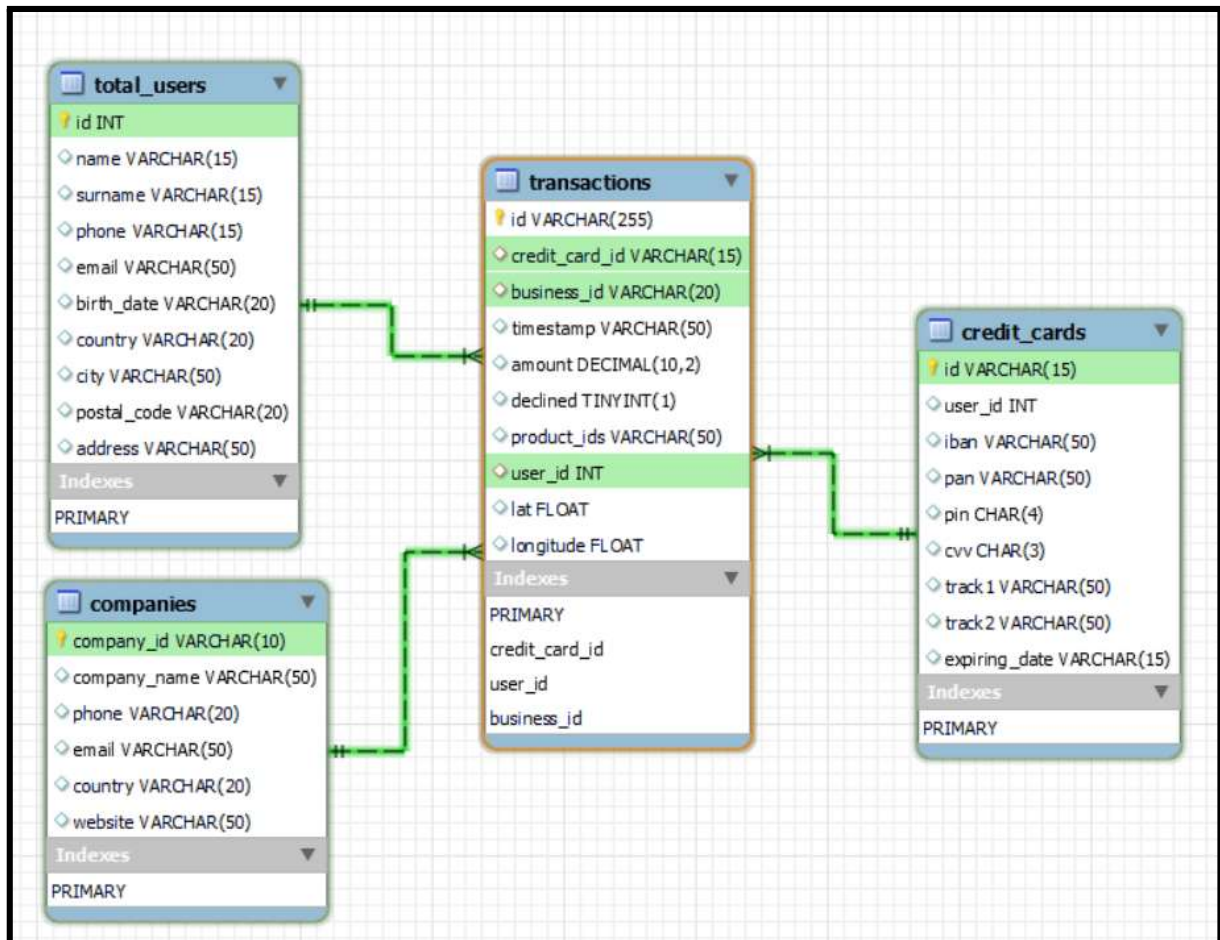
	id	credit_card_id	business_id	timestamp	amount	declined	product_ids	use
▶	02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92
	0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170
	063FBA79-99EC-66FB-29F7-25726D1764A5	CcU-2987	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275
	0668296C-CDB9-A883-76BC-2E4C4F8C8AE	CcU-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265
	06CD9AA5-9B42-D684-DDDD-A5E394FEB999	CcU-2959	b-2346	2021-10-26 23:00:01	279.93	0	43, 31	92
	07A46D48-31A3-7E87-65B9-0DA902AD109F	CcU-3225	b-2386	2021-06-28 21:11:42	340.87	1	47, 23	272
	09DE92CE-6F27-28B7-13B5-938582B38E2	CcU-3071	b-2298	2021-05-11 20:40:06	303.05	1	67, 7	275
	0A476ED9-0C13-1962-F87B-D3563924B539	CcU-4359	b-2302	2022-02-26 20:33:54	430.49	0	29, 41, 11	221
	0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	CcU-3141	b-2338	2022-03-04 14:54:35	288.81	1	19, 41, 29, 3	272
	0C7C3A33-9947-3BC1-846D-7BE3D0D17598	CcU-3309	b-2434	2021-04-10 20:58:41	103.44	1	89, 31	272
	0CE957A6-CCAA-2B7A-6839-8A4B1B324853	CcU-3435	b-2506	2022-02-02 07:29:36	428.69	1	83, 43, 73, 61	269
	0DD2E608-5C9E-D1B3-4999-B99F43AD735A	CcU-2959	b-2234	2021-04-17 05:30:17	252.47	1	7, 47, 17	275
	1017AA59-3D5F-7A4C-1992-D151A8D1FA0A	CcU-3701	b-2618	2021-11-01 01:02:11	447.11	0	37, 13	267

transactions 46 x

Output

#	Time	Action	Message
9	20:58:24	PREPARE stmt FROM 'INSERT INTO 'sprint_4'.transactions' (id', 'credit_card_id', 'business_id', 'times...	OK
10	20:58:28	DEALLOCATE PREPARE stmt	OK
11	20:58:34	SELECT * FROM transactions	587 row(s) returned

En el diagrama de la nueva base de datos "Sprint_4", se observa un modelo estrella. Este modelo está compuesto por una tabla de hechos central denominada "transactions", la cual está rodeada por tres tablas de dimensiones con información sobre usuarios, compañías y tarjetas de crédito. Cada una de estas tablas de dimensiones se conecta directamente a la tabla de hechos a través de claves foráneas, estableciendo una relación de 1 a muchos:



Exercici 1

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

Para realizar este ejercicio, primero he creado una subconsulta que cuenta todas las transacciones por usuario y filtrada por aquellos que tienen 30 o más transacciones mediante la cláusula HAVING.

Esta subconsulta también sirve para unir la tabla "total_users" con "transactions" utilizando la cláusula WHERE id IN, que combina los registros donde el campo "id" de la tabla "total_users" coincide con el campo "user_id" de "transactions" y que, además, tengan 30 o más transacciones registradas:

El resultado es que hay 4 usuarios que han realizado 30 o más transacciones:

```
111  -- EJERCICIO 1 - NIVEL 1 --
112  • SELECT *
113    FROM total_users
114   WHERE id IN
115     (SELECT user_id
116      FROM transactions
117      GROUP BY user_id
118      HAVING COUNT(id) >= 30);
```

Result Grid

	id	name	surname	phone	email	birth_date	country	city	postal_code
▶	92	Lynn	Riddle	1-387-885-4057	vitae.aliquet@outlook.edu	Sep 21, 1984	United States	Bozeman	61871
	267	Ocean	Nelson	079-481-2745	aenean@yahoo.com	Dec 26, 1991	Canada	Charlottetown	85X 3
	272	Hedwig	Gilbert	064-204-8788	sem.eget@idoud.edu	Apr 16, 1991	Canada	Tuktoyaktuk	Q4C 3
	275	Kenyon	Hartman	082-871-7248	convallis.ante.lectus@yahoo.com	Aug 3, 1982	Canada	Richmond	R8H 2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

total_users 2 x

Output

Action Output

#	Time	Action	Message
✓ 1	12:30:00	SELECT * FROM total_users WHERE id IN (SELECT user_id FROM transactions GROUP BY user_id HAVING COUNT(id) >= 30);	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:

Para realizar esta consulta, he utilizado tres tablas: "credit_cards" para obtener el IBAN, "transactions" para calcular el importe medio de las transacciones, y "companies" para filtrar por la compañía "Donec Ltd".

Primero, utilizo la instrucción SELECT para especificar los campos que deseo mostrar en la consulta. Aplico la función agregada AVG al campo amount para calcular el importe promedio de las transacciones por cada IBAN. Luego, con la cláusula GROUP BY, agrupo las filas según el número de IBAN, con lo que tenemos el importe promedio de las transacciones para cada grupo.

Combino las 3 tablas mediante "JOINS" y aplico filtro WHERE para indicar que devuelva sólo los IBAN de la compañía "Donec Ltd".

El resultado es que la empresa Donec Ltd, sólo ha usado 1 IBAN con un importe medio de transacciones de 203,72€

```
120 -- EJERCICIO 2 - NIVEL 1 --
121 • SELECT iban, company_name, round(avg(amount),2) AS MediaAmount
122 FROM credit_cards
123 JOIN transactions ON credit_cards.id = transactions.credit_card_id
124 JOIN companies ON transactions.business_id = companies.company_id
125 WHERE company_name LIKE 'Donec Ltd'
126 GROUP BY iban, company_name;
```

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

iban	company_name	MediaAmount
PT87806228135092429456346	Donec Ltd	203.72

Result 8 x

Output

Action Output

#	Time	Action	Message
✓ 1	13:07:45	SELECT iban, company_name, round(avg(amount),2) AS MediaAmount FROM credit_cards JOIN transa...	1 row(s) returned

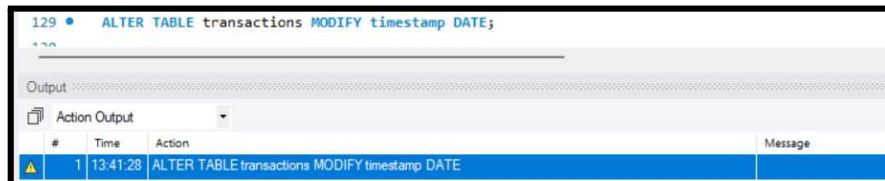
NIVELL 2

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:

Exercici 1

Quantes targetes estan actives?

Primero, es necesario modificar el formato del campo "timestamp" de la tabla "transactions" y que actualmente está como VARCHAR y necesitamos pasarlo a DATE para poder hacer operaciones de fecha:



Para obtener la nueva tabla con los datos solicitados por el enunciado, primero necesitamos crear una CTE (Common Table Expression), que actúa como una vista temporal. Usamos la cláusula WITH para definir esta CTE, nombrándola "transactions_view". Luego, creamos la nueva tabla "new_transactions" a partir de los datos en la CTE.

Uso la función ROW_NUMBER() para asignar un número de registro correlativo a cada transacción, empezando en el número 1.

Con la cláusula PARTITION BY, agrupamos las transacciones por tarjeta de crédito, reiniciando el número de registro (ROW_NUMBER) en cada salto de tarjeta.

Por ejemplo, aquí podemos ver claramente cómo funcionan ROW_NUMBER y PARTITION BY: la tarjeta de crédito 3891 tiene 17 registros, que han sido numerados del 1 al 17. Cuando se cambia a la tarjeta 3968, la numeración se reinicia y vuelve a empezar desde 1. Esto es lo que hace PARTITION BY: agrupa los registros por tarjeta y les asigna un número secuencial que se reinicia con cada nuevo grupo. Ver ejemplo, cada color es un nuevo grupo y el contador se reinicia de nuevo cada vez que cambia el color:

credit_card_id	num_registro
CcU-3981	1
CcU-3981	2
CcU-3981	3
CcU-3981	4
CcU-3981	5
CcU-3981	6
CcU-3981	7
CcU-3981	8
CcU-3981	9
CcU-3981	10
CcU-3981	11
CcU-3981	12
CcU-3981	13
CcU-3981	14
CcU-3981	15
CcU-3981	16
CcU-3981	17
CcU-3988	1
CcU-3995	1
CcU-4093	1
CcU-4093	2
CcU-4093	3
CcU-4093	4
CcU-4093	5
CcU-4100	1

Ordeno cada partición de mayor a menor según la fecha de transacción utilizando la cláusula ORDER BY DESC.

Finalmente, para obtener las 3 últimas transacciones por cada tarjeta, filtro por aquellos registros (asignados con ROW_NUMBER) que sean menores o iguales a 3 que, al estar ordenadas por fecha descendente, coincidirán con las 3 últimas transacciones de cada tarjeta.

Por último, aplico un segundo filtro con AND para ver sólo las transacciones que no fueron rechazadas (declined False).

Aparecen 292 registros que cumplen las condiciones:

```
131 -- EJERCICIO 1 - NIVEL 2 --
132 • CREATE TABLE new_transactions AS
133 WITH transactions_view AS
134 (SELECT *,
135  ROW_NUMBER() OVER (PARTITION BY credit_card_id ORDER BY timestamp DESC) AS num_registro
136  FROM transactions)
137 SELECT *
138 FROM transactions_view
139 WHERE num_registro <= 3 AND declined = 0
140 ORDER BY credit_card_id, num_registro;
```

Output

#	Time	Action	Message
1	15:23:06	CREATE TABLE new_transactions AS WITH transactions_view AS (SELECT *, ROW_NUMBER() OVE...	292 row(s) affected R...

Para visualizar las tarjetas únicas activas de la nueva tabla, que es lo que pide el ejercicio, utilizo SELECT DISTINCT(credit_card_id).

Entendiendo que estás 275 tarjetas son las que siguen activas, que en realidad son todas las de la tabla "credit_cards":

```
142 -- CUANTAS TARJETAS ÚNICAS ESTÁN ACTIVAS? --
143 • SELECT DISTINCT credit_card_id
144 FROM new_transactions;
```

Result Grid

credit_card_id
CdU-2938
CdU-2945
CdU-2952
CdU-2959
CdU-2966
CdU-2973
CdU-2980
CdU-2987
CdU-2994
CdU-3001
CdU-3008

new_transactions 3 x

Output

#	Time	Action	Message
1	15:23:06	CREATE TABLE new_transactions AS WITH transactions_view AS (SELECT *, ROW_NUMBER() OV...	292 row(s) affected R...
2	15:24:22	SELECT DISTINCT credit_card_id FROM new_transactions	275 row(s) returned

NIVELL 3

Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

La nueva tabla "products" es de dimensiones con el campo "id" como PRIMARY KEY:

```
150 -- CREACIÓ DE LA TABELLA DE DIMENSIONES "PRODUCTS" --
151 • CREATE TABLE products
152   (id INT PRIMARY KEY,
153    product_name VARCHAR(50),
154    price VARCHAR(20),
155    colour VARCHAR(20),
156    weight DECIMAL(5, 2),
157    warehouse_id VARCHAR(20));
```

Output

Action Output

#	Time	Action	Message
---	------	--------	---------

Comprobamos e importamos los datos del fichero cvs según las instrucciones mencionadas en el Nivel 1.

```
159 • SELECT * FROM products;
160
```

Result Grid

	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	#dbdbdb	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

products 13 x

Output

Action Output

#	Time	Action	Message
✓	5 22:24:29	SHOW SESSION VARIABLES LIKE 'lower_case_table_names'	OK
✓	6 22:24:29	SHOW COLUMNS FROM 'sprint_4'.products	OK
✓	7 22:25:02	PREPARE stmt FROM 'INSERT INTO 'sprint_4'.products' (id,'product_name','price','colour','weight...	OK
✓	8 22:25:03	DEALLOCATE PREPARE stmt	OK
✓	9 22:25:14	SELECT * FROM products	100 row(s) returned

Exercici 1

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

Antes de realizar esta consulta, es necesario crear una nueva tabla "by_product" y separar los datos del campo product_ids de la tabla "transactions". Actualmente, todos los productos vendidos en una única transacción están almacenados en una sola celda, separados por comas, lo que impide operar con ellos de manera adecuada. Debemos separarlos y convertir cada uno de ellos en un registro individual. También nos servirá para poder relacionar la nueva tabla con "products" y con "transactions" en una relación de muchos a uno como veremos más adelante en el nuevo diagrama:

```
161 CREATE TABLE by_product AS
162 SELECT transactions.id AS id_Transaccion, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1) AS id_Producto
163 FROM transactions
164 JOIN (SELECT 1 n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4) numbers
165 ON CHAR_LENGTH(product_ids) - CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= numbers.n - 1
166 JOIN products ON products.id = SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1);
167
```

Output

#	Time	Action	Message
1	21:41:01	CREATE TABLE by_product AS SELECT transactions.id AS id_Transaccion, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1) AS id_Producto	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Explicación detallada del código:

- **CREATE TABLE by_product AS** : Creamos la nueva tabla y le damos el nombre "by_product"
- **SELECT**: selecciono el campo id de la tabla "transactions" para poderla relacionar al modelo posteriormente.
- **SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1) AS product_id**: Esta parte del código sirve para extraer los valores de la cadena del campo "product_ids" delimitados por comas.
 - **"numbers.n"**: Este código permite iterar sobre una secuencia de números, y extrae sucesivamente cada valor dentro de la cadena "product_ids" separados por comas de la tabla "transaccions".
- **ON CHAR_LENGTH(product_ids) - CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= numbers.n - 1**: Esto es una resta entre los dos fragmentos de código:
 - **ON CHAR_LENGTH(product_ids)** el primero (en azul) nos indica el número total de caracteres que hay en la cadena "product_ids" incluyendo las comas que los separa.
 - **CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= numbers.n - 1**: Esta parte, elimina las comas de la cadena y cuenta los caracteres que quedan. La resta entre los caracteres del fragmento azul menos la del verde, nos da el número de comas que hay en la cadena. Por ejemplo, en la siguiente cadena: **1,2,3**, la longitud original es 5, y la longitud sin comas es 3: **123**, por lo que la resta es **5 - 3 = 2**. El número 2 indica cuántas comas hay en la cadena, lo que es equivalente al número de elementos en "product_ids" menos 1. Con este resultado el código sabe el número exacto de elementos hay que extraer de cada celda y convertirlos en registros separados.

Comprobación:

```
169 SELECT * FROM by_product;
```

Result Grid

id_Transaccion	id_Producto
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
0466A42E-47CF-8D24-FD01-C0B689713128	43
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	47
063FBA79-99EC-66FB-29F7-25726D1764A5	5
063FBA79-99EC-66FB-29F7-25726D1764A5	31
063FBA79-99EC-66FB-29F7-25726D1764A5	67

by_product 10 x

Output

#	Time	Action	Message
1	21:41:01	CREATE TABLE by_product AS SELECT transactions.id AS id_Transaccion, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1) AS id_Producto	1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0
2	21:41:40	SELECT * FROM by_product	1457 row(s) returned

Ahora modifico el formato del campo “product_id” de la nueva tabla de VARCHAR a INT para poder crear una clave foránea y relacionarla con la tabla “products”. Para que la relación sea correcta, ambos campos deben tener el mismo formato:

```
171 • ALTER TABLE by_product MODIFY id_Producto INT;
```

Output

Action Output

#	Time	Action	Message
✓ 1	21:38:01	ALTER TABLE by_product MODIFY id_Producto INT	1457 row(s) affected

Por último creo una clave foránea en la tabla “by_product” en el campo id_Transacciones para relacionarla con “transactions”:

```
175 • ALTER TABLE by_product ADD FOREIGN KEY (id_Transaccion) REFERENCES transactions(id);
176
```

Output

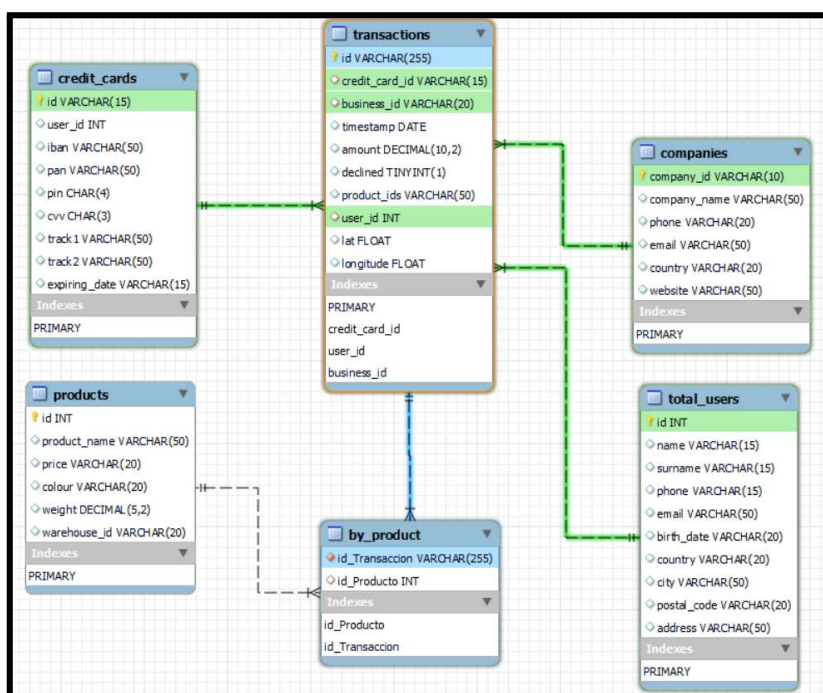
Action Output

#	Time	Action	Message
✓ 1	21:43:05	ALTER TABLE by_product ADD FOREIGN KEY (id_Transaccion) REFERENCES transactions(id)	1457 row(s) affected Records

Con estos cambios, ya tenemos relacionadas todas las tablas en nuestro modelo.

La tabla “transactions” está ahora relacionada con la nueva tabla derivada “by_product” mediante una relación de uno a muchos, donde cada registro en “transactions” puede estar asociado con múltiples registros en “by_product”.

Por otro lado, la tabla de dimensiones “products” se relaciona con la de hechos “by_product” estableciendo una relación de 1 a muchos.



Por último resuelvo el ejercicio donde tenemos que hallar cuantas veces se ha vendido cada producto mediante la función de agregación COUNT y uniendo mediante la cláusula JOIN la tabla "by product" con "products".

El resultado son 26 únicos productos vendidos en 1.457 transacciones:

```
179 • SELECT id_Producto, product_name AS Producto, COUNT(id_Producto) AS N°Ventas
180 FROM by_product
181 INNER JOIN products ON products.id = by_product.id_Producto
182 GROUP BY id_Producto, Producto
183 ORDER BY N°Ventas DESC;
```

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

	id_Producto	Producto	N°Ventas
▶	23	riverlands north	68
	67	Winterfell	68
	79	Direwolf riverlands the	66
	2	Tarly Stark	65
	43	duel	65
	47	Tully	62
	1	Direwolf Stannis	61
	17	skywalker ewok sith	61
	97	jinn Winterfell	61
	13	palpatine chewbacca	60
	53	kingblood littlefinger	58

Result 26 x

Output

Action Output

#	Time	Action	Message
✓ 1	13:25:53	SELECT id_Producto, product_name AS Producto, COUNT(id_Producto) AS N°Ventas FROM by_produ...	26 row(s) returned