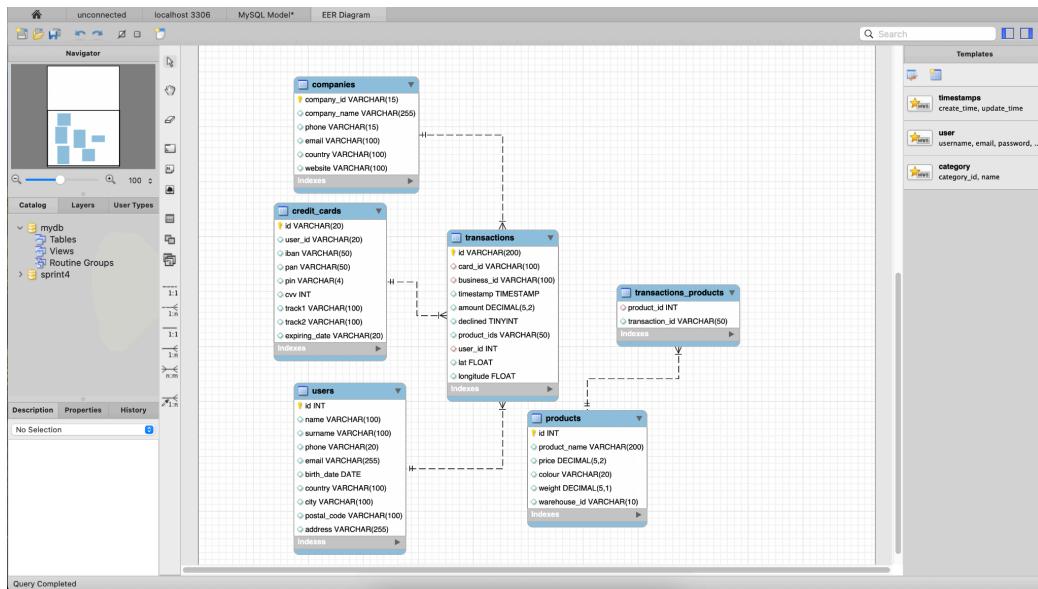


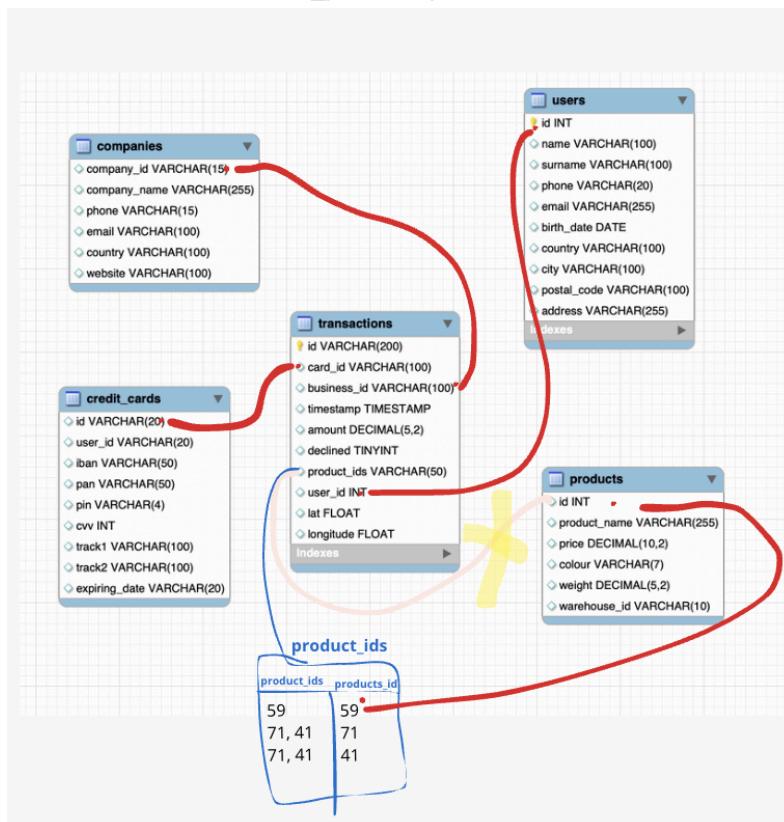
Nivell 1

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 puguis realitzar les següents consultes:

La base de datos creada va ser aquesta



Para crearla he pensado en las relaciones de la tabla de hecho transactions con la de dimensiones, comprendiendo que no era posible crear una relación directa entre la de transactions con products y que hacia falta una transactions_products para relacionar las duas.



Primero he creado la base de datos:

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'SCHEMAS', there is a tree view with 'sprint4' and 'sys' selected. The main pane displays a query editor titled 'Query 1' with the following SQL code:

```
1 • CREATE DATABASE sprint4;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

Below the code, the status bar shows 'Query Completed'. On the right side of the interface, there is a vertical toolbar with various icons and a help message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

Luego he creado las tablas de dimensiones para luego relacionar con la tabla de hechos. Empeze con Companies:

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'SCHEMAS', 'sprint4' is selected, and within it, 'Tables' is expanded to show 'companies'. The main pane displays a query editor titled 'Query 1' with the following SQL code:

```
1 • CREATE DATABASE sprint4;
2
3 -- Creando, importando y haciendo el check de la creacion de
4 -- Tabla COMPANIES: creacion, importacion y check
5 • CREATE TABLE IF NOT EXISTS companies(
6     company_id VARCHAR(15) PRIMARY KEY,
7     company_name VARCHAR(255),
8     phone VARCHAR(15),
9     email VARCHAR(100),
10    country VARCHAR (100),
11    website VARCHAR (100)
12 );
13
```

Below the code, the status bar shows 'Query Completed'. On the right side of the interface, there is a vertical toolbar with various icons and a help message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

Y importado los datos de companies.csv via codigo en la tabla, haciendo el check visualizando los datos de la tabla:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** sprint4
- Tables:** companies
- Query Editor:**

```

13
14 • LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/companies.csv'
15   INTO TABLE companies
16   FIELDS TERMINATED BY ','
17   ENCLOSED BY ''
18   LINES TERMINATED BY '\r\n'
19   IGNORE 1 LINES;
20
21 • SELECT * FROM companies;
22
    
```
- Result Grid:** Displays the imported data from the companies table.
- Action Output:**

Time	Action	Response
126 11:52:10	LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/companies.csv' INTO TABLE companies	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
127 11:52:30	SELECT * FROM companies	100 row(s) returned

Exactamente lo mismo con la tabla de Credit_Cards: crear tabla, importar data y conferir importación. Aquí tuve un problema con la terminación de líneas y tuve que sacar o '\r' de la LINES TERMINATED BY. Algunos archivos funcionaban con solo '\n' y otros necesitan ambos con '\r\n'. He pesquisado y entiendo que depende de donde se ha hecho la creación del archivo. Si el archivo ha sido creado en sistemas basados en Unix/Linux, incluidos macOS y otros sistemas derivados, se usa solo \n. Los archivos creados en sistemas Windows típicamente usan la secuencia \r\n para representar el final de una línea.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** sprint4
- Tables:** credit_cards
- Query Editor:**

```

22
23 -- Table: CREDIT_CARDS: creacion, importacion y check
24 • CREATE TABLE IF NOT EXISTS credit_cards(
25   id VARCHAR(20) PRIMARY KEY,
26   user_id VARCHAR(30),
27   iban VARCHAR(50),
28   pan VARCHAR(19),
29   pin VARCHAR(4),
30   cvv INT,
31   track1 VARCHAR(100),
32   track2 VARCHAR(100),
33   expiring_date VARCHAR (20)
34 );
35
36 • LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/credit_cards.csv'
37   INTO TABLE credit_cards
38   FIELDS TERMINATED BY ','
39   ENCLOSED BY ''
40   LINES TERMINATED BY '\r\n'
41   IGNORE 1 LINES;
42
43 • SELECT * FROM credit_cards;
44
    
```
- Result Grid:** Displays the imported data from the credit_cards table.
- Action Output:**

Time	Action	Response
128 11:53:33	CREATE TABLE IF NOT EXISTS credit_cards(id VARCHAR(20) PRIMARY KEY, user_...	0 rows(s) affected
129 11:53:37	LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/credit_cards.csv' INTO TABLE credit_cards	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0
130 11:53:40	SELECT * FROM credit_cards	275 row(s) returned

Luego la creacion de la tabla de users, que tuvo un aspecto especial porque quise insertar el dato de birthday como DATE y no como VARCHAR entonces he usado el SET para, al coger los datos, poder alterar la formatacion de string to date de esa forma: birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y'), en este orden porque esta asi en el archivo para hacer la transformacion. Tambien he juntado todos los 3 archivos de .csv de users de ca + usa + uk en una sola tabla de users.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "sprint4".
- Tables:** The "Tables" section lists "companies", "credit_cards", and "users".
- Query Editor:** The query being run is:

```

-- Tabla users creacion, importacion y check
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    surname VARCHAR(100),
    phone VARCHAR(100),
    email VARCHAR(100),
    birth_date DATE,
    country VARCHAR(100),
    city VARCHAR(100),
    postal_code VARCHAR(100),
    address VARCHAR(255)
);

LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/jets/Sprint4/users_ca.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY ''
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(@id, @name, @surname, @phone, @email, @birth_date, @country, @city, @postal_code, @address)
SET
    id = @id,
    name = @name,
    surname = @surname,
    phone = @phone,
    email = @email,
    birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y'), -- Convertiendo la fecha para el formato correcto
    country = @country,
    city = @city,
    postal_code = @postal_code,
    address = @address;

```
- Result Grid:** Shows the data loaded from users_ca.csv:

	id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum enim@protonmail.edu	1985-11-17	United States	Lowell	73544	348-7818 Sagittis St.	
2	Garrett	Mccomell	(718) 257-2412	integer.vitae.nibh@protonmail.org	1992-08-23	United States	Des Moines	59464	903 Sit Ave	
3	Claran	Harrison	(522) 598-1365	interdum.feugiat@aol.org	1998-04-29	United States	Columbus	56518	736-2063 Telus St.	
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	1989-02-18	United States	Kailua	77417	Ap #545-2244 Errat Rd.	
- Action Output:** Shows the log of operations:

Time	Action	Response
11:56:14	CREATE TABLE users	1 row(s) affected
11:56:25	LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/users_ca.csv'	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0
11:56:32	LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/users_usa.csv'	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0
11:56:37	LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/users_uk.csv'	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0
11:56:40	SELECT * FROM users	278 row(s) returned
- Code Block (Bottom):**

```

LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/users_usa.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY ''
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(@id, @name, @surname, @phone, @email, @birth_date, @country, @city, @postal_code, @address)
SET
    id = @id,
    name = @name,
    surname = @surname,
    phone = @phone,
    email = @email,
    birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y'),
    country = @country,
    city = @city,
    postal_code = @postal_code,
    address = @address;

LOAD DATA INFILE '/Users/barbarajunqueira/Desktop/RecursosDataAnalyst/Sprint4/users_uk.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY ''
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
(@id, @name, @surname, @phone, @email, @birth_date, @country, @city, @postal_code, @address)
SET
    id = @id,
    name = @name,
    surname = @surname,
    phone = @phone,
    email = @email,
    birth_date = STR_TO_DATE(@birth_date, '%b %d, %Y'),
    country = @country,
    city = @city,
    postal_code = @postal_code,
    address = @address;

```

Luego la tabla de productos:

The screenshot shows the MySQL Workbench interface with the following details:

- Administration** tab selected.
- Schemas** panel: Schema is set to **sprint4**. The **Tables** section lists **companies**, **credit_cards**, **products**, **users**, and **Views**. The **Stored Procedures** and **Functions** sections are empty. A **sys** entry is also present.
- Query Editor**:
 - Text area:

```
-- Tabla PRODUCTS: creacion, importacion y check, que tiene que ser creada posteriori de la tabla de TRANSACTION_PRODUCTS
CREATE TABLE products(
    id INT PRIMARY KEY,
    product_name VARCHAR(200),
    price DECIMAL (5, 2),
    colour VARCHAR(20),
    weight DECIMAL (5, 1),
    warehouse_id VARCHAR(10)
);

LOAD DATA INFILE '/Users/barbara junqueira/Desktop/RecursosDataAnalyst/Sprint4/products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY ''
LINES TERMINATED BY '\n';

IGNORE 1 LINES
(@id, @product_name, @price, @colour, @weight, @warehouse_id)
SET id = @id;
@product_name = @product_name,
colour = @colour,
weight = @weight,
warehouse_id = @warehouse_id;
;

SELECT * FROM products;
```
 - Status bar: 50% completion, 24:142 rows.
 - Result Grid: Shows the data from the products CSV file. The columns are **id**, **product_name**, **price**, **colour**, **weight**, and **warehouse_id**. The data includes entries like "Dwarrow Danhoz" with id 1, "Terry Stark" with id 2, and "north of Casterly" with id 7.
 - Action Output: Shows the execution history with three entries:
 - 136 12:02:16 CREATE TABLE products(id INT PRIMARY KEY, ...) 0 row(s) affected
 - 137 12:02:21 LOAD DATA INFILE '/Users/barbara junqueira/Desktop/RecursosDataAnalyst/Sprint4/products.csv' 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0
 - 138 12:02:23 SELECT * FROM products 100 row(s) returned

A context help window is open on the right side, titled "Automatic context help is disabled. Use the toolbar to manually get help for the current position or to toggle automatic help."

Luego la tabla de transactions que tiene todas las relaciones con las tablas anteriores con excepción de products, setadas en su creación:

The screenshot shows a MySQL Workbench interface with the following details:

- MySQL Model**: EER Diagram tab is selected.
- Administration**: Schemas tab is selected, showing the **sprint4** database.
- Tables**: A tree view of tables: companies, credit_cards, products, transactions, users.
- Views**, **Stored Procedures**, **Functions**, and **SYS** are also listed under Administration.
- Queries**:
 - Query 1: `CREATE TABLE transactions (id VARCHAR(200) PRIMARY KEY, card_id VARCHAR(100), business_id VARCHAR(100), timestamp DATETIME, amount DECIMAL(15, 2), declined TINYINT, product_id VARCHAR(50), user_id INT, lat FLOAT, longitude FLOAT, FOREIGN KEY (business_id) REFERENCES companies(company_id), FOREIGN KEY (card_id) REFERENCES credit_cards(card_id), FOREIGN KEY (user_id) REFERENCES users(user_id))`
 - Query 2: `LOAD DATA INFILE '/Users/barbara junqueira/Desktop/RecursosDataAnalyst/sprint4/transactions.csv' INTO TABLE transactions FIELDS TERMINATED BY ',' LINES TERMINATED BY '\r\n';`
 - Query 3: `SELECT * from transactions;`
- Result Grid**: Shows the results of the last query with 1169 rows.
- Object Info** and **Session** tabs are visible.
- Schema: sprint4** is selected.
- Transactions**: A table showing transaction details with columns: id, card_id, business_id, timestamp, amount, declined, product_ids, user_id, lat, longitude.
- Action Output**: A table showing the execution of three SQL statements with columns: Time, Action, Response.
- Context... Snippet**: A sidebar with the message "Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help."

Pensando en establecer las conexiones de la tabla de hechos con la de dimensiones me he dado cuenta de que la tabla de transactions no podria unirse con la de productos porque aunque la de transactions tuviera los productos comprados, estaban estos en un "pack" que era el product_ids. entonces hice la creacion de una tabla llamada transactions_products con el objetivo de separar los products_ids(transactions_id en la tabla en question) en product_id individual que se podría conectar con la tabla de productos. Como se trata de una tabla de unión entre la tabla de hechos y también la de productos, todas las relaciones en ambos lados se dan de muchos a muchos. por eso he creado un id para identificar de manera unica cada linea.

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The current schema is "sprint4".
- Tables:** The table "transactions_products" is selected.
- Script Editor:** The SQL code for creating the table and inserting data is visible. The table has columns: id_transaction_products (INT AUTO_INCREMENT PRIMARY KEY), transaction_id (INT), and product_id (VARCHAR(50)). The insert query uses a complex UNION ALL SELECT statement to split the comma-separated product_ids into individual rows.
- Result Grid:** The data grid shows 14 rows of data, mapping transaction IDs to various product IDs.
- Action Output:** The log shows three actions: creating the table, inserting data, and selecting from it.
- Help:** A context help panel on the right provides information about automatic context help being disabled.

PD: al principio quise añadir una relación de many-to-many de transactions a transactions_products pero no he podido por el error 1822. como he leído que no tiene sentido crear esa relación en cuestión por esa misma cuestión de ser many-to-many dejé sin la relación como se puede ver en el pantallazo de diagrama.

The screenshot shows the MySQL Workbench interface with the following details:

- Action Output:** The log shows an attempt to alter the transactions table to add a foreign key constraint named "transactions_ibfk_4" that references the transactions_products table. The operation fails with Error Code: 1822, indicating a missing index for the constraint.

ROSER M'HA EXPLICAT COM FER-HO MILLOR AMB L'ID DE TRANSACCIÓ I HO HE FET:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Apple, MySQLWorkbench, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Title Bar:** Untitled - MySQL Workbench.
- Schemas Tab:** Administration, Schemas, schemas (localhost 3308), MySQL Model*, EER Diagram.
- Query Editor:** Shows a complex UNION ALL SELECT statement for creating a table named transactions_products. The statement includes joins on transaction_id and product_id, and uses SUBSTRING_INDEX and CHAR_LENGTH functions to handle varying lengths of product_ids.
- Result Grid:** Displays the data from the transactions_products table, which contains 28 rows. The columns are transaction_id and product_id.
- Object Info:** No object selected.
- Action Output:** Shows the execution time (12:55:40) and the query: "SELECT * FROM transactions_products".
- Session:** Response: 1457 row(s) returned.
- Toolbar:** Includes icons for Home, New, Open, Save, Print, Copy, Paste, Find, Refresh, Undo, Redo, and Help.
- Help:** A context-sensitive help panel on the right side provides instructions for using the toolbar to toggle automatic help.

Compreendi que si yo tenía el product_ids estos se podrían repetir porque dos empresas/usuarios diferentes podrían hacer una compra igual. Pero el ID de la transacció de la tabla de transactions siempre sería único y esa es una buena manera de hacerlo mejor!

Luego quise mejorar la selección de elementos para “cuantos haya” y no solo 4 (porque pude hacer el check manual pero entendiendo que habrá archivos en los cuales no podré hacerlo quiero garantizar que todos los elementos de la columna product_ids entre dentro de la nueva tabla, entonces aplique una recursividad que lo que hace es coger la cantidad de elementos que tiene la columna y coletar los datos esa cantidad X de veces.

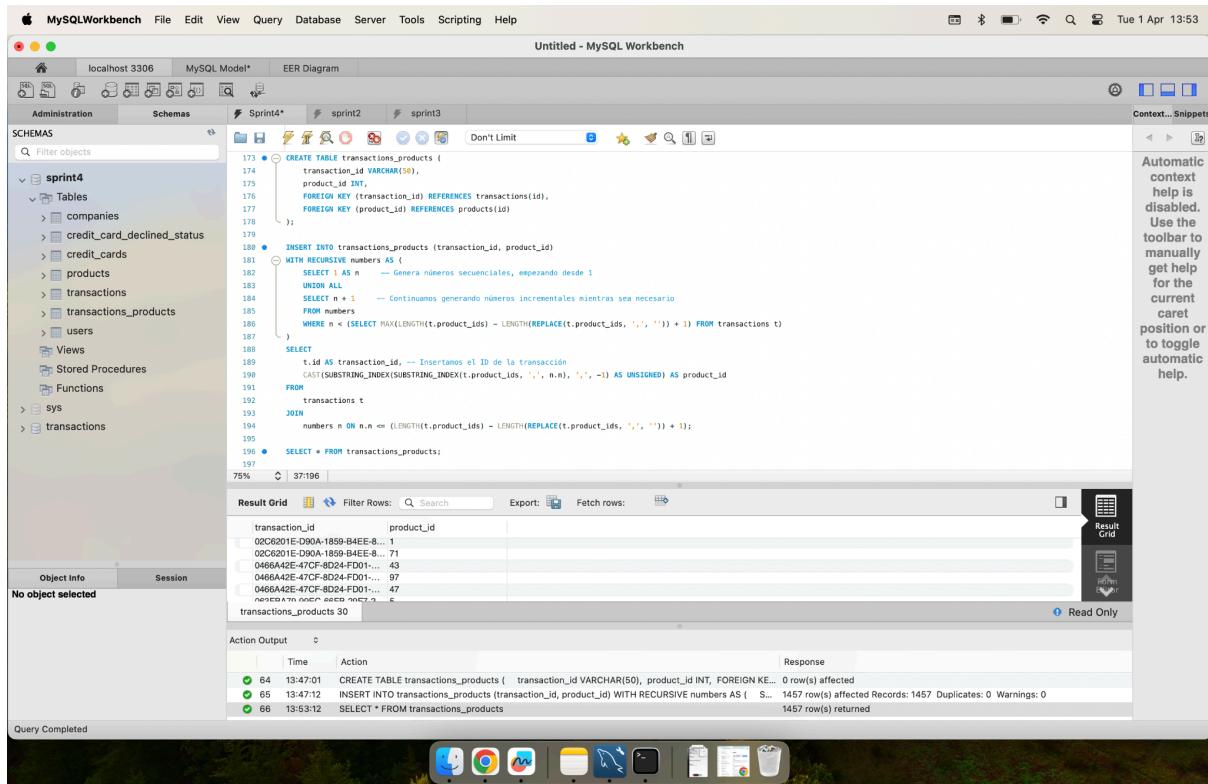
Explicacion

WHERE n < (SELECT MAX(LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1) FROM transactions t)

1. LENGTH(t.product_ids): Calcula la longitud total de la cadena product_ids. Por ejemplo, si tienes product_ids = '1,2,3,5,9,12', esto devolvería 11 (porque hay 11 caracteres en total, incluyendo las comas).
 2. REPLACE(t.product_ids, ',', ''): Elimina todas las comas en la cadena, dejando solo los números. Usando el mismo ejemplo de product_ids = '1,2,3,5,9,12', después de aplicar REPLACE, obtienes '1235912' (longitud de 7).
 3. LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')): Esto da como resultado el número de comas en la cadena. En nuestro ejemplo, hay 5 comas, por lo que el resultado de la resta es 5.
 4. MAX es usado para calcular el número máximo de productos en una transacción.
 5. + 1: Finalmente, sumamos 1, porque el número de elementos es una coma menos que el número de valores. En nuestro caso, como hay 5 comas, sumando 1 nos da 6, que es el número total de elementos separados por comas en la lista.
 6. La condición WHERE n < (SELECT LENGTH(...) + 1) asegura que la recursión se detenga cuando el número n llegue a más de los valores que hay en la lista de product_ids. Si la cadena tiene 6 elementos, solo se generarán números hasta el 6, y no más.

```
CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n.n), ',', -1) AS UNSIGNED) AS product_id
```

La columna de product_id será rellenada por la columna de product_ids pero separando cada una que este separada por virgula hasta el enésimo valor. El -1 es para coger la última entrada de valor (vuelve la casilla anterior). El CAST convierte la subcadena extraída en un número entero sin signo (UNSIGNED). Si fueran números negativos usaría el SIGNED para mantener la señal



The screenshot shows the MySQL Workbench interface with the EER Diagram tab selected. A query window displays the creation of the `transactions_products` table:

```

CREATE TABLE transactions_products (
    transaction_id VARCHAR(50),
    product_id INT,
    FOREIGN KEY (transaction_id) REFERENCES transactions(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

INSERT INTO transactions_products (transaction_id, product_id)
WITH RECURSIVE numbers AS (
    SELECT 1 AS n  -- Genera números secuenciales, empezando desde 1
    UNION ALL
    SELECT n + 1  -- Continuamos generando números incrementales mientras sea necesario
    FROM numbers
    WHERE n < (SELECT MAX(LENGTH(t.product_ids)) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1) FROM transactions t
)
SELECT
    t.id AS transaction_id, -- Insertamos el ID de la transacción
    CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n), ',', '-1') AS UNSIGNED) AS product_id
FROM
    transactions t
JOIN
    numbers n ON n.n <= (LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1);
SELECT * FROM transactions_products;

```

The Result Grid shows the generated data:

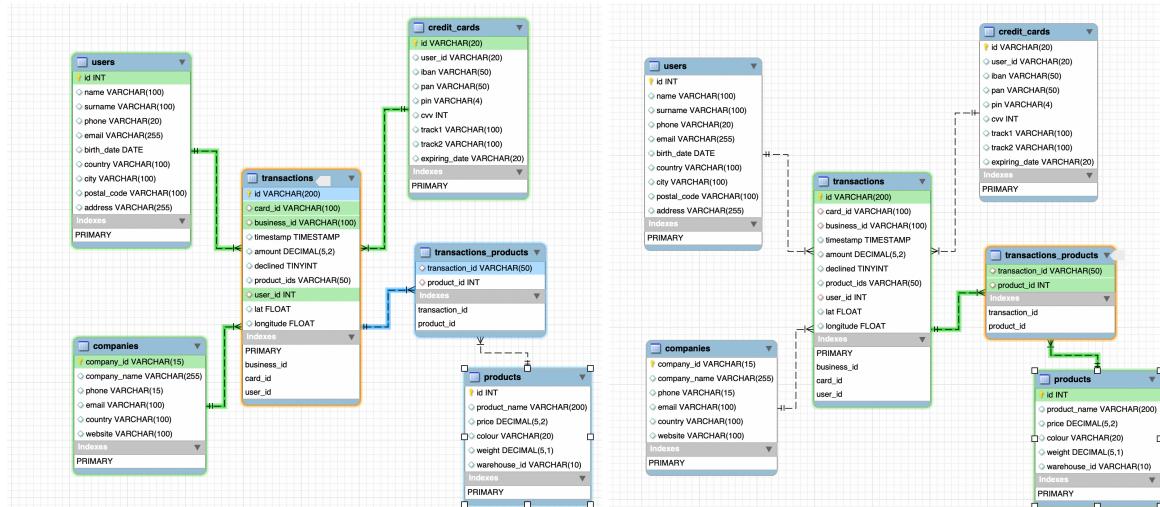
transaction_id	product_id
02C6201E-090A-1859-B4EE-8...	1
02C6201E-090A-1859-B4EE-8...	71
046842E-47CF-BD24-FD01...	43
046842E-47CF-BD24-FD01...	97
046842E-47CF-BD24-FD01...	47

The Action Output pane shows the execution log:

- CREATE TABLE transactions_products (transaction_id VARCHAR(50), product_id INT, FOREIGN KEY (transaction_id) REFERENCES transactions(id), FOREIGN KEY (product_id) REFERENCES products(id)) affected: 0 rows(s)
- INSERT INTO transactions_products (transaction_id, product_id) WITH RECURSIVE numbers AS (SELECT 1 AS n -- Genera números secuenciales, empezando desde 1 UNION ALL SELECT n + 1 -- Continuamos generando números incrementales mientras sea necesario FROM numbers WHERE n < (SELECT MAX(LENGTH(t.product_ids)) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1) FROM transactions t) SELECT t.id AS transaction_id, -- Insertamos el ID de la transacción CAST(SUBSTRING_INDEX(SUBSTRING_INDEX(t.product_ids, ',', n), ',', '-1') AS UNSIGNED) AS product_id FROM transactions t JOIN numbers n ON n.n <= (LENGTH(t.product_ids) - LENGTH(REPLACE(t.product_ids, ',', '')) + 1); affected: 1457 rows(s) affected Records: 1457 Duplicates: 0 Warnings: 0
- SELECT * FROM transactions_products affected: 1457 row(s) returned

Entonces ahora tiene sentido hacer la relación de

1 a muchos de `transactions.id` a `transactions_products.transaction_id` y también
muchos a 1 de `transactions_products.product_id` a `products.id` 😊😊😊



- Exercici 1

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

```
209      -- Nivell 1 Exercici 1
210      -- Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.
211 •  SELECT u.name as userName, u.surname as userSurname, count(business_id) as transaccions
212     FROM transactions t
213     JOIN users u ON t.user_id = u.id
214     GROUP BY userName, userSurname
215     HAVING count(business_id) > 30
216     ORDER BY count(business_id) DESC;
217
```

The screenshot shows a database query results interface. At the top, there is a code editor window with the query from step 211. Below it is a results grid titled "Result Grid" with columns "userName", "userSurname", and "transaccions". The data shows four rows: Hedwig (76), Ocean (52), Kenyon (48), and Lynn (39). A sidebar on the right has icons for "Result Grid" and "Form Editor". Below the results grid is a "Result 4" section. At the bottom, there is an "Action Output" table with columns "Time", "Action", "Response", and "Duration / Fetch Time". One row is visible: "10 10:35:27 SELECT u.name as userName, u.surname as userSurname, count(business_id) as transaccions FROM t... 4 row(s) returned 0.0058 sec / 0.00002...".

- Exercici 2

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

```
218      -- Nivell 1 Exercici 2
219      -- Mostra la mitjana d'amount per IBAN de les targetes de crèdit a la companyia Donec Ltd, utilitza almenys 2 taules.
220      -- pre consulta para entender si seria un avg alto o bajo, y se puede calcular manualmente. tendria que ser (364,61 + 42,82) / 2 = 203,715
221 •  SELECT *
222     FROM credit_cards cc
223     JOIN transactions t ON t.card_id = cc.id
224     JOIN companies c ON t.business_id = c.company_id
225     WHERE c.company_name = "Donec Ltd";
226
227     -- querry en question:
228 •  SELECT avg(amount) as avgAmount, company_name as companyName, card_id as iban
229     FROM transactions t
230     JOIN companies c ON t.business_id = c.company_id
231     WHERE c.company_name = "Donec Ltd"
232     GROUP BY iban, companyName;
233
```

The screenshot shows a database query results interface. At the top, there is a code editor window with the query from step 228. Below it is a results grid titled "Result Grid" with columns "avgAmount", "companyName", and "iban". The data shows one row: 203.715000, Donec Ltd, CcU-2973. A sidebar on the right has icons for "Result Grid" and "Form Editor". Below the results grid is a "Result 12" section. At the bottom, there is an "Action Output" table with columns "Time", "Action", "Response", and "Duration / Fetch Time". One row is visible: "20 10:50:18 SELECT avg(amount) as avgAmount, company_name as companyName, card_id as iban FROM transac... 1 row(s) returned 0.0016 sec / 0.00002...".

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:

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'sprint4' schema is selected. Inside 'Tables', there is a newly created table named 'credit_card_declined_status'. The 'Script' tab at the top contains the SQL code for creating the table and inserting data. The 'Result Grid' tab shows the resulting data with 275 rows. The 'Action Output' tab at the bottom displays the execution history of the queries.

```
-- Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions van ser declinades
CREATE TABLE credit_card_declined_status AS
WITH lastThreeTransactions AS (
    SELECT t.card_id, t.declined,ROW_NUMBER() OVER (PARTITION BY t.card_id ORDER BY t.timestamp DESC) AS numRow
    FROM transactions t
)
SELECT card_id,
CASE
    WHEN COUNT(CASE WHEN declined = 1 THEN 1 END) = 3 THEN 'declined'
    ELSE 'active'
END AS status
FROM lastThreeTransactions
WHERE numRow <= 3
GROUP BY card_id;
SELECT * FROM credit_card_declined_status;
```

card_id	status
CU-2945	active
CU-2952	active
CU-2954	active
CU-2958	active
CU-2973	active
CU-2980	active
CU-2987	active
CU-2994	active
CU-3001	active
CU-3010	active
CU-3015	active
CU-3022	active
CU-3029	active
CU-3035	active
CU-3043	active
CU-3052	active
CU-3057	active
CU-3064	active

Action Output

Time	Action	Response	Duration / Fetch Time
26 10:59:37	CREATE TABLE credit_card_declined_status AS WITH lastThreeTransactions AS (SELECT t.card_id,... 275 row(s) affected Records: 275 Duplicates: 0 War...	0.022 sec
27 10:59:39	SELECT * FROM credit_card_declined_status	275 row(s) returned	0.0014 sec / 0.00008...

Exercici 1

Quantes targetes estan actives?

The screenshot shows the MySQL Workbench interface. The 'Script' tab at the top contains a single SQL query to count active credit cards. The 'Result Grid' tab shows the result of the query. The 'Action Output' tab at the bottom displays the execution history of the query.

```
SELECT count(status) as tarjetasActivas
FROM credit_card_declined_status
WHERE status = 'active';
```

tarjetasActivas
275

Action Output

Time	Action	Response
38 11:59:38	SELECT count(status) as tarjetasActivas FROM credit_card_declined_status WHERE status = 'active'	1 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 tabla ya habia sido creada a principio:

The screenshot shows the MySQL Workbench interface. In the top navigation bar, 'File', 'Edit', 'View', 'Query', 'Database', 'Server', 'Tools', 'Scripting', and 'Help' are visible. The main area displays a script window with SQL code for creating a table 'transactions_products'. The code includes a LOAD DATA INFILE command to import data from 'products.csv', a CREATE TABLE statement for 'transactions_products', and a complex SELECT query to map product IDs from 'products' to 'transaction_id' in 'transactions_products'. Below the script, a 'Result Grid' shows the resulting data with columns 'id_transaction_product', 'product_id', and 'transaction_id'. The 'Object Info' panel on the left shows the table structure with columns 'id_transaction_products' (PK), 'product_id' (AI PK), and 'transaction_id' (varchar(50)). The status bar at the bottom indicates 'Query Completed'.

Exercici 1

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

The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL code:

```
-- Necessitem conèixer el nombre de vegades que s'ha venut cada producte.
269 • SELECT product_id AS produpte, COUNT(product_id) AS nombreVegadesVenut
270 FROM transactions_products tp
271 JOIN transactions t ON tp.transaction_id = t.id
272 WHERE declined = 0
273 GROUP BY produpte
274 ORDER BY produpte;
```

The 'Result Grid' below the code shows the results of the query, with columns 'produpte' and 'nombreVegadesVenut'. The data includes rows for product IDs 1 through 17, with their respective counts. The status bar at the bottom indicates 'Query Completed'.