

# ***SPRINT 4v2***

---

**Edoardo Brega**

# ★ NIVELL 1

## → exercici 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:

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

### ❖ creo database y tabla transactions

```
1 • create database sprint_4_v2; use sprint_4_v2;
2 • CREATE TABLE transactions (
3     id varchar (200)PRIMARY KEY,
4     card_id varchar (100) not null,
5     business_id varchar (100),
6     timestamp varchar (100),
7     amount double,
8     declined int,
9     product_ids varchar (100),
10    user_id varchar(100),
11    lat double,
12    longitude double,
13    index company_key (business_id),    index card_key (card_id),    index product_ids_key (product_ids),
14    index timestamp_key (timestamp),    index user_key(user_id),    index declined_key(declined));
```

Output

#	Time	Action	Message
✓ 1	18:29:57	create database sprint_4_v2	1 row(s) affected
✓ 2	18:29:57	use sprint_4_v2	0 row(s) affected
✓ 3	18:29:57	CREATE TABLE transactions (id varchar (200)PRIMARY KEY, card_id...	0 row(s) affected

### ➤ código:

```
create database sprint_4_v2; use sprint_4_v2;
CREATE TABLE transactions (
    id varchar (200)PRIMARY KEY,
    card_id varchar (100) not null,
    business_id varchar (100),
    timestamp varchar (100),
    amount double,
    declined int,
    product_ids varchar (100),
    user_id varchar(100),
    lat double,
    longitude double,
    index company_key (business_id),    index card_key (card_id),    index product_ids_key
(product_ids),
    index timestamp_key (timestamp),    index user_key(user_id),    index
declined_key(declined));
```

❖ creo tabla companies

```
20
21 • CREATE TABLE companies (
22     company_id varchar (200) PRIMARY KEY,
23     company_name varchar (100),
24     phone varchar (100),
25     email varchar (100),
26     country varchar (100),
27     website varchar (100),
28     INDEX id_key (company_id));
29
```

Output			
Action Output			
#	Time	Action	Message
✓ 1	18:33:47	CREATE TABLE companies ( company_id varchar (200) PRIMARY KEY, ...	0 row(s) affected

➤ código:

```
CREATE TABLE companies (
    company_id varchar (200) PRIMARY KEY,
    company_name varchar (100),
    phone varchar (100),
    email varchar (100),
    country varchar (100),
    website varchar (100),
    INDEX id_key (company_id));
```

❖ creo tabla credit\_cards

```
33
34 ● ○ CREATE TABLE credit_cards (
35     id varchar (100) PRIMARY KEY,
36     user_id varchar (100),
37     iban varchar (100),
38     pan varchar (100),
39     pin int,
40     cvv varchar (100),
41     track1 varchar (100),
42     track2 varchar (100),
43     expiring_date varchar (100),
44     INDEX id_key (id));
```

Output			
Action Output			
#	Time	Action	Message
✓ 1	18:36:23	CREATE TABLE credit_cards (id varchar (100) PRIMARY KEY, user_j...	0 row(s) affected

➤ código:

```
CREATE TABLE credit_cards (
    id varchar (100) PRIMARY KEY,
    user_id varchar (100),
    iban varchar (100),
    pan varchar (100),
    pin int,
    cvv varchar (100),
    track1 varchar (100),
    track2 varchar (100),
    expiring_date varchar (100),
    INDEX id_key (id));
```

❖ creo tabla products

```
49
50 • CREATE TABLE products (
51     id varchar (100) PRIMARY KEY,
52     product_name varchar (100),
53     price varchar (100),
54     colour varchar (100),
55     weight double,
56     warehouse_id varchar (100),
57     index id_key(id));
58
```

Output			
Action Output			
#	Time	Action	Message
1	18:38:52	CREATE TABLE products (id varchar (100) PRIMARY KEY, product_n...	0 row(s) affected

➤ código:

```
CREATE TABLE products (
    id varchar (100) PRIMARY KEY,
    product_name varchar (100),
    price varchar (100),
    colour varchar (100),
    weight double,
    warehouse_id varchar (100),
    index id_key(id));
```

❖ **creo tabla users**

```
62 • CREATE TABLE users (  
63     id VARCHAR(100) PRIMARY KEY,  
64     name VARCHAR(100),  
65     surname VARCHAR(100),  
66     phone VARCHAR(100),  
67     email VARCHAR(100),  
68     birth_date VARCHAR(100),  
69     country VARCHAR(100),  
70     city VARCHAR(100),  
71     postal_code VARCHAR(100),  
72     address VARCHAR(100),  
73     index id_key(id));  
74
```

Output

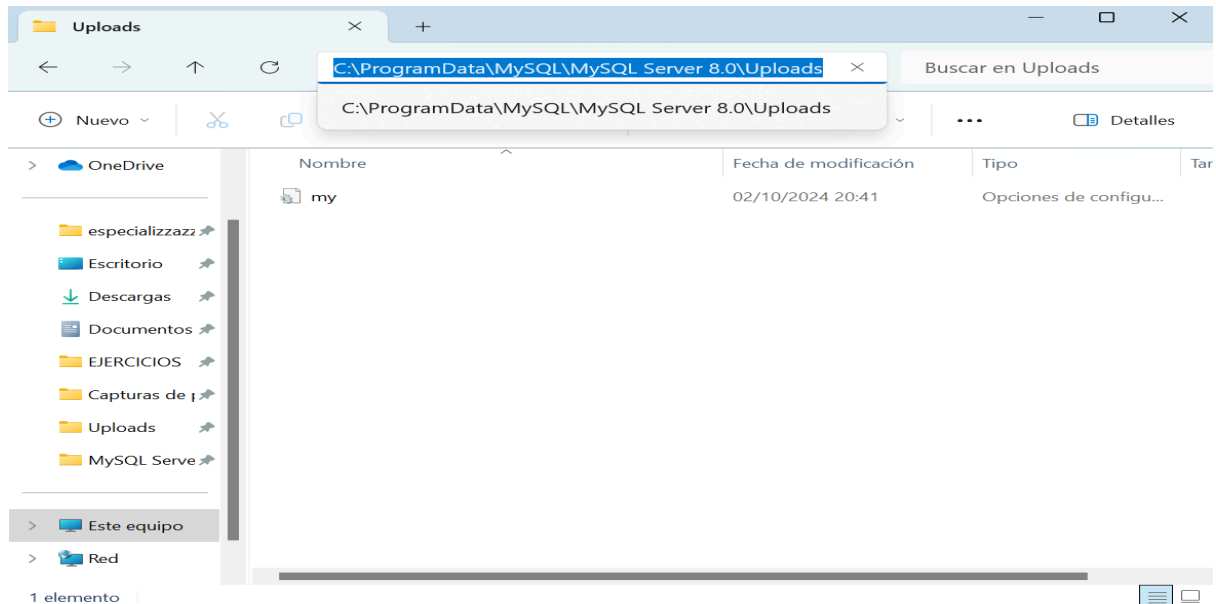
Action Output

#	Time	Action	Message
1	19:11:48	CREATE TABLE users ( id VARCHAR(100) PRIMARY KEY, name V...	0 row(s) affected

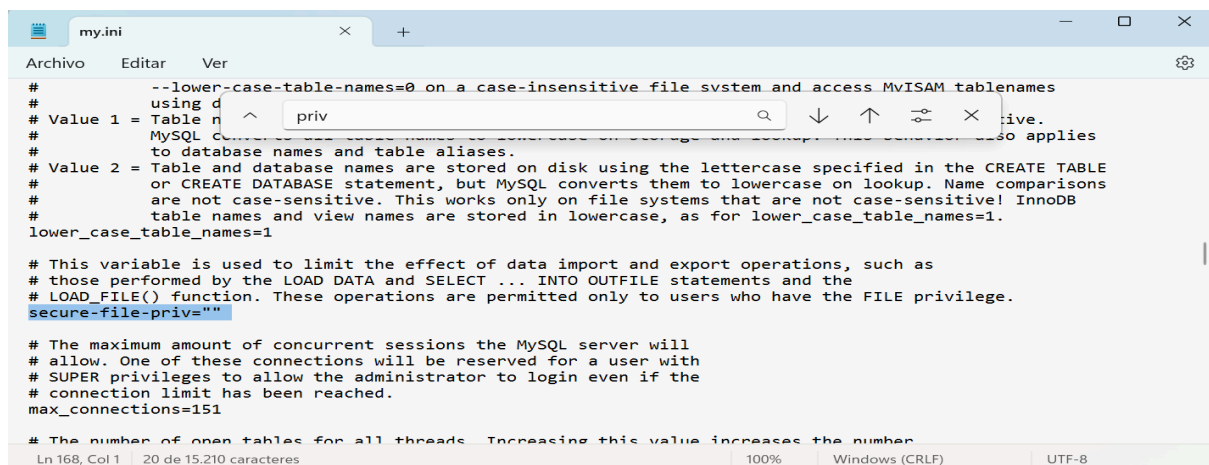
➤ código:

```
CREATE TABLE users (  
    id VARCHAR(100) PRIMARY KEY,  
    name VARCHAR(100),  
    surname VARCHAR(100),  
    phone VARCHAR(100),  
    email VARCHAR(100),  
    birth_date VARCHAR(100),  
    country VARCHAR(100),  
    city VARCHAR(100),  
    postal_code VARCHAR(100),  
    address VARCHAR(100),  
    index id_key(id));
```

- ❖ busco el file my.ini para desactivar la seccion 'secure\_file\_priv' para poder insertar los datos directamente con codigo



- ❖ dejo la seccion 'secure\_file\_priv' en blanco



❖ compruebo que no haya ningun valor en la seccion modificada

```
1 • SHOW VARIABLES LIKE 'secure_file_priv';
2
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Variable_name	Value		
secure_file_priv			

Result 3	×		
Output			
Action Output			
#	Time	Action	Message
1	21:00:13	SHOW VARIABLES LIKE 'secure_file_priv'	1 row(s) returned

❖ copio los archivos descargados en la carpeta del database creada

MySQL	MySQL Server 8.0	Data	sprint_4_v2	Buscar en sprint_4_v2
Ordenar	Ver	...	Detalles	
Nombre	Fecha de modificación	Tipo	Tamaño	
companies.ibd	12/10/2024 18:33	Archivo IBD	128 KB	
credit_cards.ibd	12/10/2024 18:49	Archivo IBD	144 KB	
products.ibd	12/10/2024 18:38	Archivo IBD	128 KB	
transactions.ibd	12/10/2024 18:49	Archivo IBD	208 KB	
users.ibd	12/10/2024 18:41	Archivo IBD	128 KB	
companies	28/09/2024 20:10	OpenOffice.org XML ...	11 KB	
users_usa	28/09/2024 20:10	OpenOffice.org XML ...	19 KB	
users_uk	28/09/2024 20:10	OpenOffice.org XML ...	7 KB	
users_ca	28/09/2024 20:10	OpenOffice.org XML ...	9 KB	
transactions	28/09/2024 20:10	OpenOffice.org XML ...	72 KB	
products	28/09/2024 20:10	OpenOffice.org XML ...	5 KB	
credit_cards	28/09/2024 20:10	OpenOffice.org XML ...	41 KB	



## ❖ inserto los valores manualmente

```
1 • LOAD DATA INFILE 'transactions.csv' INTO TABLE transactions FIELDS TERMINATED BY ';' LINES TERMINATED BY '\n' IGNORE 1 LINES;
2 • LOAD DATA INFILE 'companies.csv' INTO TABLE companies FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES;
3 • LOAD DATA INFILE 'credit_cards.csv' INTO TABLE credit_cards FIELDS TERMINATED BY ';' LINES TERMINATED BY '\n' IGNORE 1 LINES;
4 • LOAD DATA INFILE 'products.csv' INTO TABLE products FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES;
5 • LOAD DATA INFILE 'users_ca.csv' INTO TABLE users FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
6 LINES TERMINATED BY '\r' IGNORE 1 LINES;
7 • LOAD DATA INFILE 'users_usa.csv' INTO TABLE users FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
8 LINES TERMINATED BY '\r' IGNORE 1 LINES;
9 • LOAD DATA INFILE 'users_uk.csv' INTO TABLE users FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
10 LINES TERMINATED BY '\r' IGNORE 1 LINES;
11 • UPDATE users SET id = REPLACE(id, '\n', '');
```

Output				
Action Output				
#	Time	Action	Message	
✓ 1	20:06:48	LOAD DATA INFILE 'transactions.csv' INTO TABLE transactions FIELDS ...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 2	20:06:48	LOAD DATA INFILE 'companies.csv' INTO TABLE companies FIELDS T...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 3	20:06:48	LOAD DATA INFILE 'credit_cards.csv' INTO TABLE credit_cards FIELDS...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 4	20:06:48	LOAD DATA INFILE 'products.csv' INTO TABLE products FIELDS TERM...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 5	20:06:48	LOAD DATA INFILE 'users_ca.csv' INTO TABLE users FIELDS TERMIN...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 6	20:06:48	LOAD DATA INFILE 'users_usa.csv' INTO TABLE users FIELDS TERMI...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 7	20:06:48	LOAD DATA INFILE 'users_uk.csv' INTO TABLE users FIELDS TERMIN...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0	
✓ 8	20:06:48	UPDATE users SET id = REPLACE(id, '\n', '');	275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0	

### ➤ código:

```
LOAD DATA INFILE 'transactions.csv' INTO TABLE transactions FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'companies.csv' INTO TABLE companies FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'credit_cards.csv' INTO TABLE credit_cards FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'products.csv' INTO TABLE products FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'users_ca.csv' INTO TABLE users FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'users_usa.csv' INTO TABLE users FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r' IGNORE 1 LINES;
```

```
LOAD DATA INFILE 'users_uk.csv' INTO TABLE users FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\r' IGNORE 1 LINES;
```

```
UPDATE users SET id = REPLACE(id, '\n', '');
```

➤ explicacion:

`LOAD DATA INFILE '...' INTO TABLE ..`

Es para seleccionar el file con los valores y la tabla donde subirlos

`FIELDS TERMINATED BY '...'`

Indico por que caracter estan separados los campos del file seleccionado

`LINES TERMINATED BY '...'`

Indico por que caracter estan separadas las lineas del file seleccionado

`IGNORE ... LINES`

Indico cuantas lineas no se tienen que subir a la tabla

`OPTIONALLY ENCLOSED BY '...'`

Es para no tomar los caracteres dentro del caracter seleccionado como si fueran delimitadores de campo.

Para los 3 archivos de utentes ha sido necesario tambien eliminar manualmente la ultima linea blanca de estos files.

Y una vez creada la nueva tabla con los valores de los tres archivos, modifiko la columna 'id' para eliminar este caracter que molesta la relacion con la tabla 'transactions'.

`(UPDATE users SET id = REPLACE(id, '\n', " ");)`

## ❖ añadido FOREIGN KEY

```
1 • ALTER TABLE credit_cards ADD CONSTRAINT fk_credit_card_user FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE;
2
3 • ALTER TABLE transactions ADD CONSTRAINT fk_transaction_user FOREIGN KEY (user_id) REFERENCES users(id);
4
5 • ALTER TABLE transactions ADD CONSTRAINT fk_transaction_card FOREIGN KEY (card_id) REFERENCES credit_cards(id);
6
7 • ALTER TABLE transactions ADD CONSTRAINT fk_transaction_business FOREIGN KEY (business_id) REFERENCES companies(company_id);
```

Output				
Action Output				
#	Time	Action	Message	
✓ 1	20:14:10	ALTER TABLE credit_cards ADD CONSTRAINT fk_credit_card_user FO...	275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0	
✓ 2	20:14:10	ALTER TABLE transactions ADD CONSTRAINT fk_transaction_user FO...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	
✓ 3	20:14:10	ALTER TABLE transactions ADD CONSTRAINT fk_transaction_card FO...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	
✓ 4	20:14:10	ALTER TABLE transactions ADD CONSTRAINT fk_transaction_business...	587 row(s) affected Records: 587 Duplicates: 0 Warnings: 0	

➤ código:

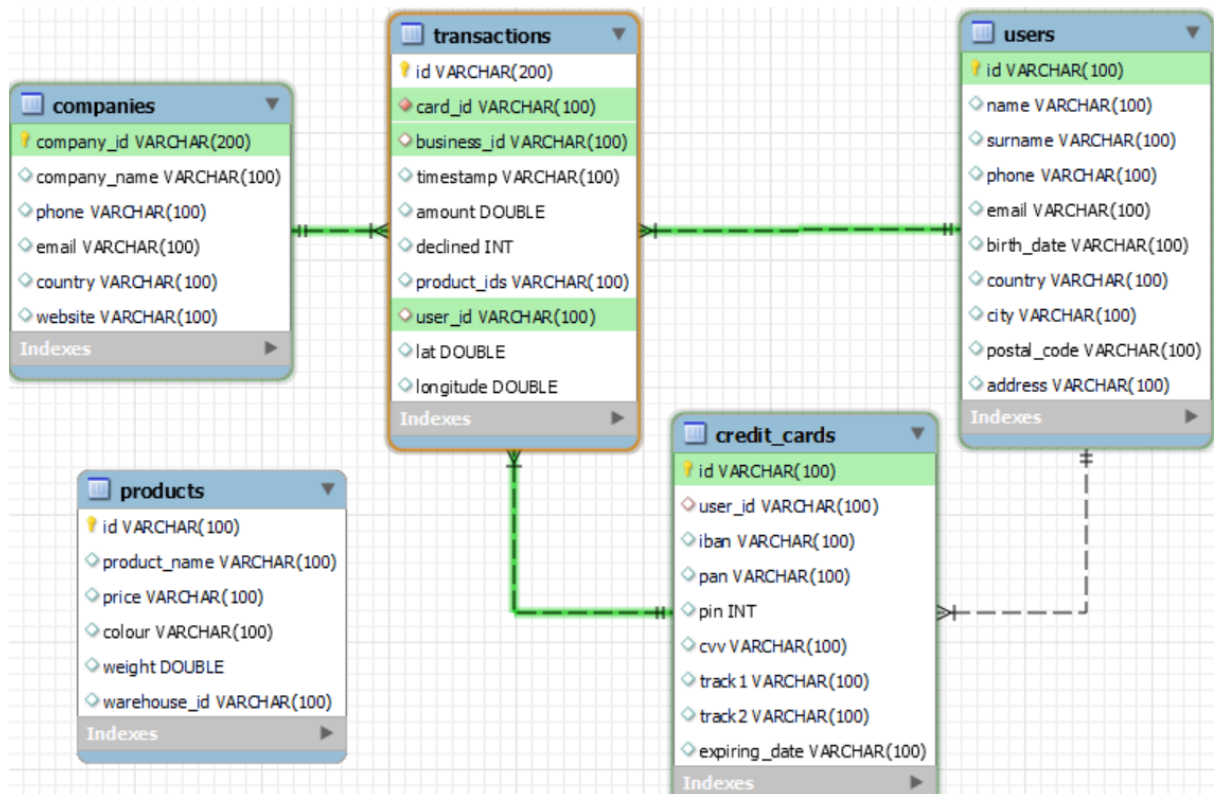
```
ALTER TABLE credit_cards ADD CONSTRAINT fk_credit_card_user FOREIGN KEY
(user_id) REFERENCES users(id) ON DELETE CASCADE;
```

```
ALTER TABLE transactions ADD CONSTRAINT fk_transaction_user FOREIGN KEY
(user_id) REFERENCES users(id);
```

```
ALTER TABLE transactions ADD CONSTRAINT fk_transaction_card FOREIGN KEY
(card_id) REFERENCES credit_cards(id);
```



```
ALTER TABLE transactions ADD CONSTRAINT fk_transaction_business FOREIGN KEY
(business_id) REFERENCES companies(company_id);
```

➤ muestro el modelo creado



❖ selecciono los usuarios con mas de 30 transacciones

```
1 • SELECT DISTINCT users.id, users.name, users.surname FROM users
2   JOIN credit_cards ON users.id = credit_cards.user_id
3   JOIN transactions ON transactions.card_id = credit_cards.id
4   WHERE credit_cards.id IN (
5       SELECT card_id FROM transactions
6       WHERE declined = 0
7       GROUP BY card_id
8       HAVING COUNT(*) > 30
9   )
10  ORDER BY users.id;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	id	name	surname
▶	267	Ocean	Nelson
	272	Hedwig	Gilbert
	92	Lynn	Riddle

Result 8 x			
Output			
Action Output			
#	Time	Action	Message
✓ 1	20:27:37	SELECT DISTINCT users.id, users.name, users.surname FROM users JO...	3 row(s) returned

➤ código:

```
select distinct users.id, users.name, users.surname
from users
join credit_cards on users.id = credit_cards.user_id
join transactions on transactions.card_id = credit_cards.id
where credit_cards.id in (
    select card_id
    from transactions
    where declined = 0
    group by card_id
    having count(*) > 30
)
order by users.id;
```

➤ explicación:

En la subconsulta:

```
select card_id from transactions where declined=0 group by card_id having count(*) >30
```

Filtro los id de las tarjetas de credito que tienen mas de 30 transacciones no declinadas.  
En la consulta principal relaciono los id de los usuarios con los id de las tarjetas filtradas y con los nombres y apellidos de los usuarios desde la tabla 'users'  
No utilizo alias para las tablas para dejar el código más comprensible al momento de la explicación.

## → exercici 2

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

❖ selecciono la media por iban de la compañía Donec Ltd

```
10
11 • select credit_cards.iban 'iban Donec Ltd',avg(transactions.amount)'mitjana d'amount' from credit_cards
12 join transactions on transactions.card_id=credit_cards.id
13 join companies on transactions.business_id=companies.company_id
14 where transactions.declined = 0 and companies.company_name='Donec Ltd'
15 group by credit_cards.iban
16 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

iban Donec Ltd	mitjana d'amount
PT87806228135092429456346	42.82

Result 16 | Result 17 x

Output

Action Output

#	Time	Action	Message
✓ 1	22:26:31	select credit_cards.iban,avg(transactions.amount) from credit_cards join transactions ...	1 row(s) returned
✓ 2	22:27:29	select distinct credit_cards.user_id 'usuaris amb més de 30 transaccions' from credit_...	3 row(s) returned
✓ 3	22:27:30	select credit_cards.iban 'iban Donec Ltd',avg(transactions.amount)'mitjana d'amount' ...	1 row(s) returned

- código:  
select credit\_cards.iban 'iban Donec Ltd',avg(transactions.amount)'mitjana d'amount' from credit\_cards  
join transactions on transactions.card\_id=credit\_cards.id  
join companies on transactions.business\_id=companies.company\_id  
where transactions.declined = 0 and companies.company\_name='Donec Ltd'  
group by credit\_cards.iban;
- explicación:  
Selecciono los iban desde la tabla 'credit\_cards', la media de los importes de las transacciones no declinadas desde la tabla 'transactions'.  
Relaciono los resultados con la tabla 'companies' para filtrar los valores de la compañía 'Donec Ltd'.

## ★ NIVELL 2

### → exercici 1

→ 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:  
Quantes targetes estan actives?

- ❖ creo la tabla que refleccione si las tarjetas estan activas (utilizo la tabla que ya existe por motivos de comodidad. Si tuviera que dejar esta envariada crearia una gemela como ya hice en el ejercicio 1 y añadiría la columna requerida)

```
1 • alter table transactions add activa varchar(10) default 'si' after card_id;
2 • with
3   recent_transactions as ( select card_id, declined, row_number()over(partition by card_id order by timestamp desc)as rn from transactions),
4   declined_cards as ( select card_id from recent_transactions
5                       where rn <= 3 group by card_id having sum(declined) > 2)
6
7   update transactions set activa = 'no' where card_id in (select card_id from declined_cards);
8 • select*from transactions;
```

id	card_id	activa	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
02C6201E-D90A-1859-B4EE-88D2986D3B02	CdU-2938	si	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92	81.9184589824	-12.5275561984
0466A42E-47CF-8D24-FD01-C0B689713128	CdU-4219	si	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.9694885888	-117.5251835904
063FBA79-99EC-66FB-29F7-25726D1764A5	CdU-2987	si	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.222680576	-129.049879552
0668296C-CD89-A883-76BC-2E4C4F8C8AE	CdU-3743	si	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265	-34.3593055232	-100.555928064
06CD9AA5-9B42-D684-DDDD-A5E394FBA99	CdU-2959	si	b-2346	2021-10-26 23:00:01	279.93	0	43, 31	92	33.7381445632	158.298210304

transactions 11 x

Output

#	Time	Action	Message	Duration / Fetch
1	12:01:41	alter table transactions add activa varchar(10) default 'si' after card_id	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.015 sec
2	12:01:41	with recent_transactions as(select card_id,declined,row_number()over(...	0 row(s) affected Rows matched: 0 Changed: 0 Warnings: 0	0.000 sec
3	12:01:41	select*from transactions	587 row(s) returned	0.000 sec / 0.000 sec

#### ➤ código:

```
alter table transactions add activa varchar(10) default 'si' after card_id;
with
  recent_transactions as(select card_id,declined, row_number()over(partition by
card_id order by timestamp desc)as rn from transactions),
  declined_cards as(select card_id from recent_transactions
  where rn <= 3 group by card_id having sum(declined) > 2)
update transactions set activa = 'no' where card_id in (select card_id from declined_cards);
```

#### ➤ explicación:

ALTER TABLE transactions ADD activa varchar (10) default 'si' AFTER card\_id;

-Creo la nueva columna donde aparecerà el estado de la tarjeta.

WITH..

-Creo dos tablas temporaneas:

en la primera asigno un valor numerico a las transacciones segun la fecha, dando los valores mas bajos a las transacciones mas recientes,

en la segunda filtro las tres mas recientes y sumo su valor de declined para verificar si las tres ultimas transacciones han sido declinadas.

UPDATE transactions SET activa = 'no' where card\_id in (...)

-Modifico la columna creada segun la condiciones requeridas:

si el card\_id esta en la lista creada precedentemente con los card\_id che tienen las ultimas tre transacciones declinadas, entonces cambia e valor a 'no'.

### ❖ cuento las tarjetas activas

```
1 • select count(*) 'targetes activos'
2   from (
3     select distinct card_id
4     from transactions where activa='si')x;
5
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
targetes activos			
▶ 275			

Result 12 x			
Output			
Action Output			
#	Time	Action	Message
✓ 16	21:40:32	select * from transactions	587 row(s) returned
✓ 17	21:40:41	select count(*) 'targetes activos' from (select distinct card_id from transa...	1 row(s) returned

- código:  
select count(\*) 'targetes activos' from (select distinct card\_id from transactions where activa='si')x;
- explicación:  
Cuento el número de tarjetas distintas que tengan valor 'si' a la columna que define si están activas.

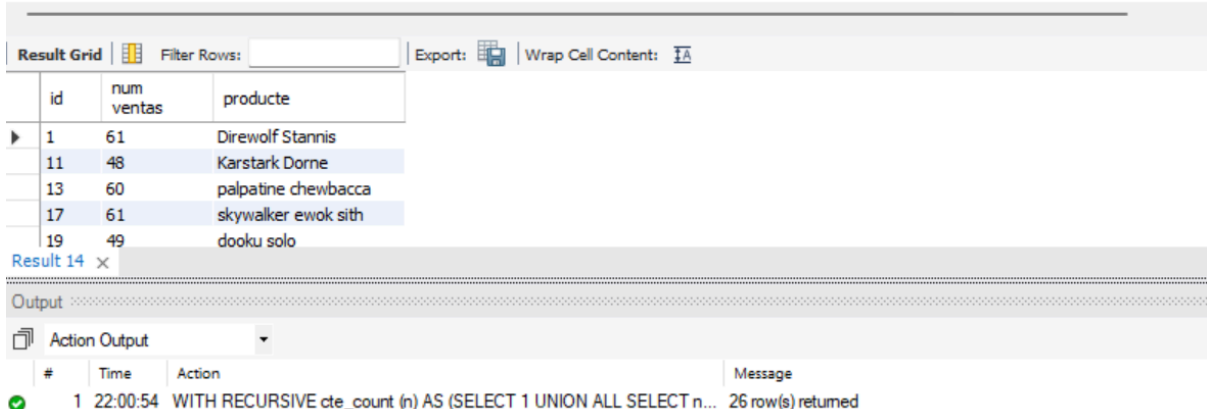


## ★ NIVELL 3

### → exercici 1

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

```
1 • WITH RECURSIVE cte_count (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte_count WHERE n < 10)
2
3 select x.idprod 'id', count(*) 'num ventas',products.product_name'producte' from(
4     SELECT TRIM( BOTH FROM SUBSTRING_INDEX( SUBSTRING_INDEX(product_ids, ',', n) ,',',-1) )AS idprod
5     FROM transactions
6     JOIN cte_count cnt WHERE cnt.n <= LENGTH(product_ids) -LENGTH(REPLACE(product_ids,',','')) +1) x
7     JOIN products ON x.idprod = products.id
8     group by idprod
9     order by idprod;
```



	id	num ventas	producte
▶	1	61	Direwolf Stannis
	11	48	Karstark Dorne
	13	60	palpatine chewbacca
	17	61	skywalker ewok sith
	19	49	dooku solo

Result 14 ×

Output

Action Output

#	Time	Action	Message
✓ 1	22:00:54	WITH RECURSIVE cte_count (n) AS (SELECT 1 UNION ALL SELECT n...	26 row(s) returned

➤ codigo:

```
WITH RECURSIVE cte_count (n) AS (  
    SELECT 1 UNION ALL SELECT n + 1  
    FROM cte_count  
    WHERE n < 10)
```

```
select x.idprod 'id', count(*) 'num ventas',products.product_name'producte'  
from(  
    SELECT TRIM( BOTH FROM SUBSTRING_INDEX(  
        SUBSTRING_INDEX(product_ids, ',', n) ,',',-1) )AS idprod  
    FROM transactions  
    JOIN cte_count cnt WHERE cnt.n <=  
    LENGTH(product_ids)-LENGTH(REPLACE(product_ids,',','')) +1) x  
    JOIN products ON x.idprod = products.id  
    group by idprod  
    order by idprod;
```

➤ explicación:

```
-WITH RECURSIVE cte_count (n) AS (  
    SELECT 1,3 UNION ALL SELECT n + 1,n*n  
    FROM cte_count  
    WHERE n < 10)
```

-En esta parte del código creo una tabla (Common Table Expression) con números secuenciales que necesitare para relacionarla con la consulta siguiente.  
Suponiendo que ninguna sola compra pase los 10 productos.

```
-select x.idprod 'id', count(*) 'num ventas',products.product_name'producte'
```

-Selecciono el ID del producto, cuántas veces se ha vendido y el nombre relacionado por el id con la tabla 'products'

```
-TRIM(BOTH FROM..)
```

-Elimina los espacios en blanco alrededor del valor extraído.

```
-SUBSTRING_INDEX(product_ids, ',', n)
```

-Esta función toma la cadena product\_ids y devuelve la subcadena hasta el n-ésimo separador.

Por una compra de 3 productos "1,25,74" y n=2 SUBSTRING\_INDEX(product\_ids, ',', 2) devolverá '1,25'.

```
-SUBSTRING_INDEX( SUBSTRING_INDEX(product_ids, ',', n) ,',',-1)
```

-Esta función toma la subcadena obtenida en el 'Substring\_index' anterior y devuelve el último elemento después del último separador.

Continuando con el ejemplo anterior, SUBSTRING\_INDEX('1,25', ',', -1) devolverá '25'.

```
-JOIN cte_count cnt
```

-Une la tabla transactions con la CTE cte\_count para contar todos los ID en la columna 'product\_ids'

```
-LENGTH(product_ids)
```

-Es la longitud de caracteres de toda la cadena

```
-REPLACE(product_ids,',','')
```

-Elimina las comas

```
-LENGTH(REPLACE(product_ids,',',''))
```

-Es la longitud de la cadena sin las comas

```
-WHERE cnt.n <= LENGTH(product_ids) - LENGTH(REPLACE(product_ids, ',', '')) + 1
```

-La resta entre la longitud de todos los caracteres menos la longitud de caracteres sin las comas nos da como resultado el número de comas -1.

Con el +1 al final obtengo el número de comas (que es igual al número de productos-1) y me aseguro que 'n' no exceda el número de productos en 'product\_ids'.

```
-JOIN products ON x.idprod = products.id
```

-Relaciono el id de producto filtrado por la subquery generado por la tabla temporanea 'x' con el id de la tabla 'products'