

# ***SPRINT 4***

---

**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 tablas

```
1 • create database sprint_4_visual; use sprint_4_visual;
2 • CREATE TABLE transactions (id varchar (200),card_id varchar (100) not null,business_id varchar (100),timestamp varchar (100),amount double,declined int,
3   product_ids varchar (100),user_id varchar (100),lat double,longitude double,index company_key (business_id),index card_key (card_id),
4   index product_ids_key (product_ids),index user_key(user_id),PRIMARY KEY id (id));
5 • CREATE TABLE companies (company_id varchar (200) not null,company_name varchar (100),phone varchar (100),email varchar (100),country varchar (100),
6   website varchar (100),PRIMARY KEY company_id (company_id),INDEX id_key (company_id),FOREIGN KEY (company_id) REFERENCES transactions(business_id));
7 • CREATE TABLE credit_cards (id varchar (100) not null,user_id varchar (100),iban varchar (100),pan varchar (100),pin int,cvv varchar (100),track1 varchar (100),
8   track2 varchar (100),expiring_date varchar (100),PRIMARY KEY (id),INDEX id_key (id),FOREIGN KEY (id) REFERENCES transactions(card_id));
9 • CREATE TABLE products (id varchar (100) not null,product_name varchar (100),price varchar (100),colour varchar (100),weight double,warehouse_id varchar (100),
10  PRIMARY KEY (id),index id_key(id));
11 • CREATE TABLE user_ca (id varchar (100) not null,name varchar (100),surname varchar (100),phone varchar (100),email varchar (100),birth_date varchar (100),
12  country varchar (100),city varchar (100),postal_code varchar(100),address varchar (100),PRIMARY KEY (id),index id_key(id));
13 • CREATE TABLE user_usa (id varchar (100) not null,name varchar (100),surname varchar (100),phone varchar (100),email varchar (100),birth_date varchar (100),
14  country varchar (100),city varchar (100),postal_code varchar(100),address varchar (100),PRIMARY KEY (id),index id_key(id));
15 • CREATE TABLE user_uk (id varchar (100) not null,name varchar (100),surname varchar (100),phone varchar (100),email varchar (100),birth_date varchar (100),
16  country varchar (100),city varchar (100),postal_code varchar(100),address varchar (100),PRIMARY KEY (id),index id_key(id));
```

### ❖ executo comprobando que no haya errores

```
1 • create database sprint_4_visual; use sprint_4_visual;
2 • CREATE TABLE transactions (id varchar (200),card_id varchar (100) not null,business_id varchar (100),timestamp varchar (100),amount double,declined int,
3   product_ids varchar (100),user_id varchar (100),lat double,longitude double,index company_key (business_id),index card_key (card_id),
4   index product_ids_key (product_ids),index user_key(user_id),PRIMARY KEY id (id));
5 • CREATE TABLE companies (company_id varchar (200) not null,company_name varchar (100),phone varchar (100),email varchar (100),country varchar (100),
6   website varchar (100),PRIMARY KEY company_id (company_id),INDEX id_key (company_id),FOREIGN KEY (company_id) REFERENCES transactions(business_id));
7 • CREATE TABLE credit_cards (id varchar (100) not null,user_id varchar (100),iban varchar (100),pan varchar (100),pin int,cvv varchar (100),track1 varchar (100),
8   track2 varchar (100),expiring_date varchar (100),PRIMARY KEY (id),INDEX id_key (id),FOREIGN KEY (id) REFERENCES transactions(card_id));
9 • CREATE TABLE products (id varchar (100) not null,product_name varchar (100),price varchar (100),colour varchar (100),weight double,warehouse_id varchar (100),
10  PRIMARY KEY (id),index id_key(id));
```

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
1	23:18:38	create database sprint_4_visual	1 row(s) affected	0.000 sec
2	23:18:38	use sprint_4_visual	0 row(s) affected	0.000 sec
3	23:18:38	CREATE TABLE transactions (id varchar (200),card_id varchar (100) not null,business_id varchar (100),times...	0 row(s) affected	0.031 sec
4	23:18:38	CREATE TABLE companies (company_id varchar (200) not null,company_name varchar (100),phone varch...	0 row(s) affected	0.031 sec
5	23:18:38	CREATE TABLE credit_cards (id varchar (100) not null,user_id varchar (100),iban varchar (100),pan varchar...	0 row(s) affected	0.031 sec
6	23:18:38	CREATE TABLE products (id varchar (100) not null,product_name varchar (100),price varchar (100),colour v...	0 row(s) affected	0.032 sec
7	23:18:39	CREATE TABLE user_ca (id varchar (100) not null,name varchar (100),surname varchar (100),phone varcha...	0 row(s) affected	0.031 sec
8	23:18:39	CREATE TABLE user_usa (id varchar (100) not null,name varchar (100),surname varchar (100),phone varch...	0 row(s) affected	0.031 sec
9	23:18:39	CREATE TABLE user_uk (id varchar (100) not null,name varchar (100),surname varchar (100),phone varcha...	0 row(s) affected	0.031 sec

➤ código:

```
create database sprint_4_visual; use sprint_4_visual;
```

```
CREATE TABLE transactions (id varchar (200),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 user_key(user_id),PRIMARY KEY id (id));
```

```
CREATE TABLE companies (company_id varchar (200) not null,company_name varchar  
(100),phone varchar (100),email varchar (100),country varchar (100),  
website varchar (100),PRIMARY KEY company_id (company_id),INDEX id_key  
(company_id),FOREIGN KEY (company_id) REFERENCES transactions(business_id));
```

```
CREATE TABLE credit_cards (id varchar (100) not null,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),PRIMARY KEY (id),INDEX id_key  
(id),FOREIGN KEY (id) REFERENCES transactions(card_id));
```

```
CREATE TABLE products (id varchar (100) not null,product_name varchar (100),price varchar  
(100),colour varchar (100),weight double,warehouse_id varchar (100),  
PRIMARY KEY (id),index id_key(id));
```

```
CREATE TABLE user_ca (id varchar (100) not null,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),PRIMARY KEY (id),index id_key(id));
```

```
CREATE TABLE user_usa (id varchar (100) not null,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),PRIMARY KEY (id),index id_key(id));
```

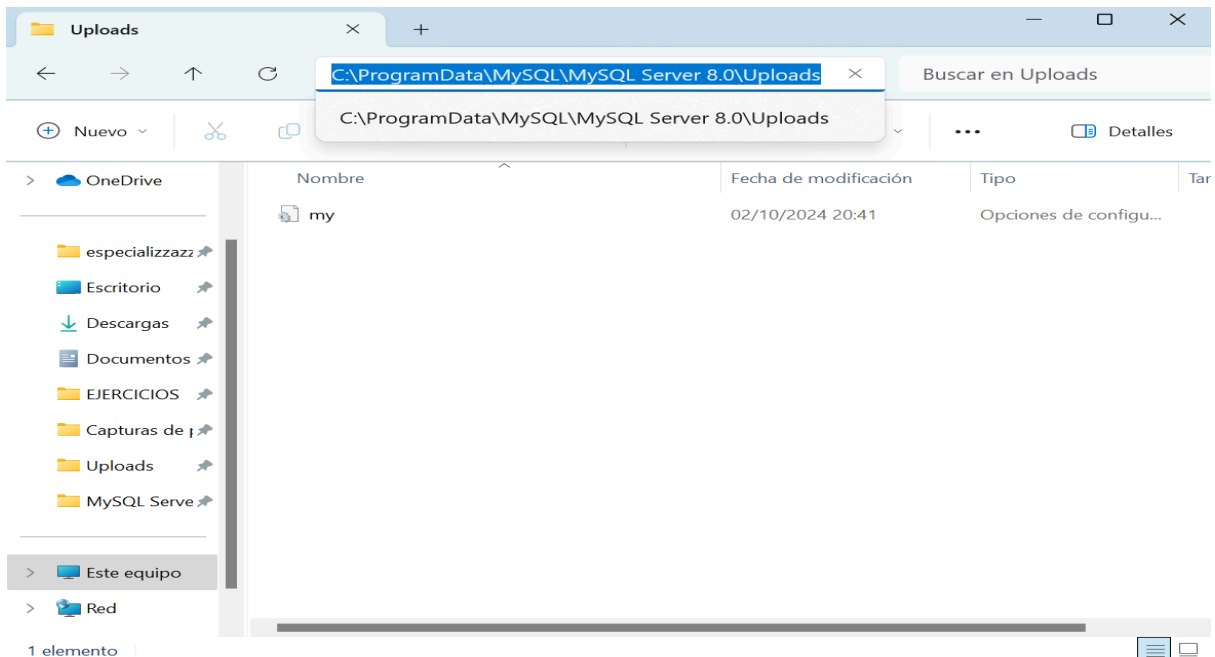
```
CREATE TABLE user_uk (id varchar (100) not null,name varchar (100),surname varchar  
(100),phone varchar (100),email varchar (100),birth_date varchar (100),  
country varchar (100),city varchar (100),postal_code varchar(100),address varchar  
(100),PRIMARY KEY (id),index id_key(id));
```

➤ explicacion:

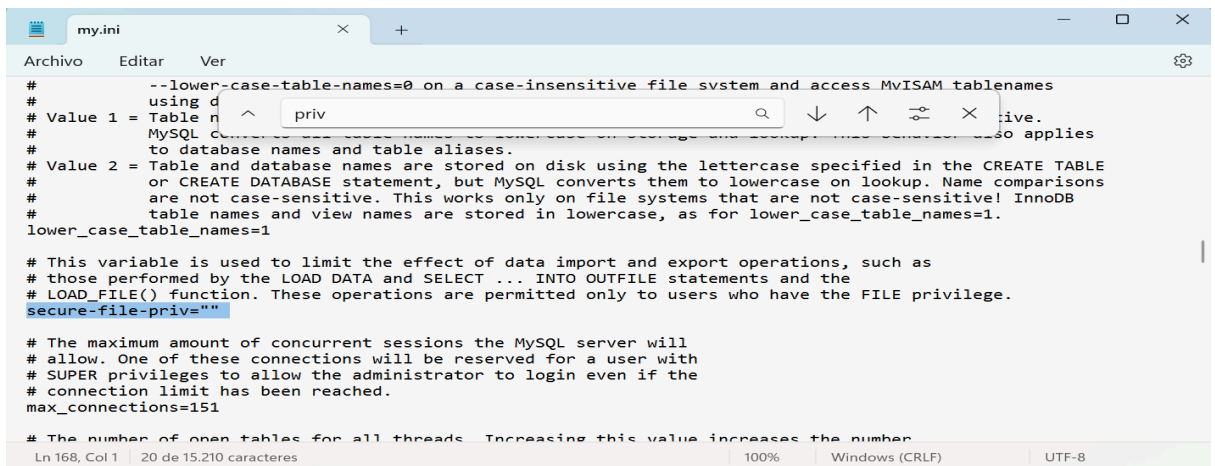
Primero creo el database y luego creo las tablas, empezando por la tabla de hechos que será 'transactions'.

No añado la 'foreign key' en las tablas 'products, user\_ca, user\_usa, user\_uk' porque los valores en los archivos darían error al intentar de insertarlos manualmente.

- ❖ 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	
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

C:\ProgramData\MySQL\MySQL Server 8.0\Data\sprint_4_visual				
C:\Users\Usuario\Downloads\EJERCICIOS				
Nombre	Fecha de modificación	Tipo	Tamaño	
 companies.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 credit_cards.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 products.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 transactions.ibd	02/10/2024 23:18	Archivo IBD	176 KB	
 user_ca.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 user_uk.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 user_usa.ibd	02/10/2024 23:18	Archivo IBD	128 KB	
 companies.csv	28/09/2024 20:10	Archivo CSV	11 KB	
 credit_cards.csv	28/09/2024 20:10	Archivo CSV	41 KB	
 products.csv	28/09/2024 20:10	Archivo CSV	5 KB	
 transactions.csv	28/09/2024 20:10	Archivo CSV	72 KB	
 users_ca.csv	02/10/2024 23:05	Archivo CSV	9 KB	
 users_uk.csv	02/10/2024 22:58	Archivo CSV	7 KB	
 users_usa.csv	02/10/2024 23:00	Archivo CSV	19 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 user_ca FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''
6		LINES TERMINATED BY '\r' IGNORE 1 LINES;
7	•	LOAD DATA INFILE 'users_usa.csv' INTO TABLE user_usa FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''
8		LINES TERMINATED BY '\r' IGNORE 1 LINES;
9	•	LOAD DATA INFILE 'users_uk.csv' INTO TABLE user_uk FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ''
10		LINES TERMINATED BY '\r' IGNORE 1 LINES;

#	Time	Action	Message	Duration / Fetch
✓ 1	23:28:09	LOAD DATA INFILE 'transactions.csv' INTO TABLE transactions FIELDS TERMINATED BY ',' LINES TER...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0	0.062 sec
✓ 2	23:28:09	LOAD DATA INFILE 'companies.csv' INTO TABLE companies FIELDS TERMINATED BY ',' LINES TERMI...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
✓ 3	23:28:09	LOAD DATA INFILE 'credit_cards.csv' INTO TABLE credit_cards FIELDS TERMINATED BY ',' LINES TER...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0	0.031 sec
✓ 4	23:28:09	LOAD DATA INFILE 'products.csv' INTO TABLE products FIELDS TERMINATED BY ',' LINES TERMINAT...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
✓ 5	23:28:09	LOAD DATA INFILE 'users_ca.csv' INTO TABLE user_ca FIELDS TERMINATED BY ',' OPTIONALLY EN...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
✓ 6	23:28:09	LOAD DATA INFILE 'users_usa.csv' INTO TABLE user_usa FIELDS TERMINATED BY ',' OPTIONALLY E...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0	0.015 sec
✓ 7	23:28:09	LOAD DATA INFILE 'users_uk.csv' INTO TABLE user_uk FIELDS TERMINATED BY ',' OPTIONALLY ENC...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec

➤ 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 user_ca FIELDS TERMINATED BY ','  
OPTIONALLY ENCLOSED BY '"'   
LINES TERMINATED BY '\r' IGNORE 1 LINES;
```

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

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

➤ 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.

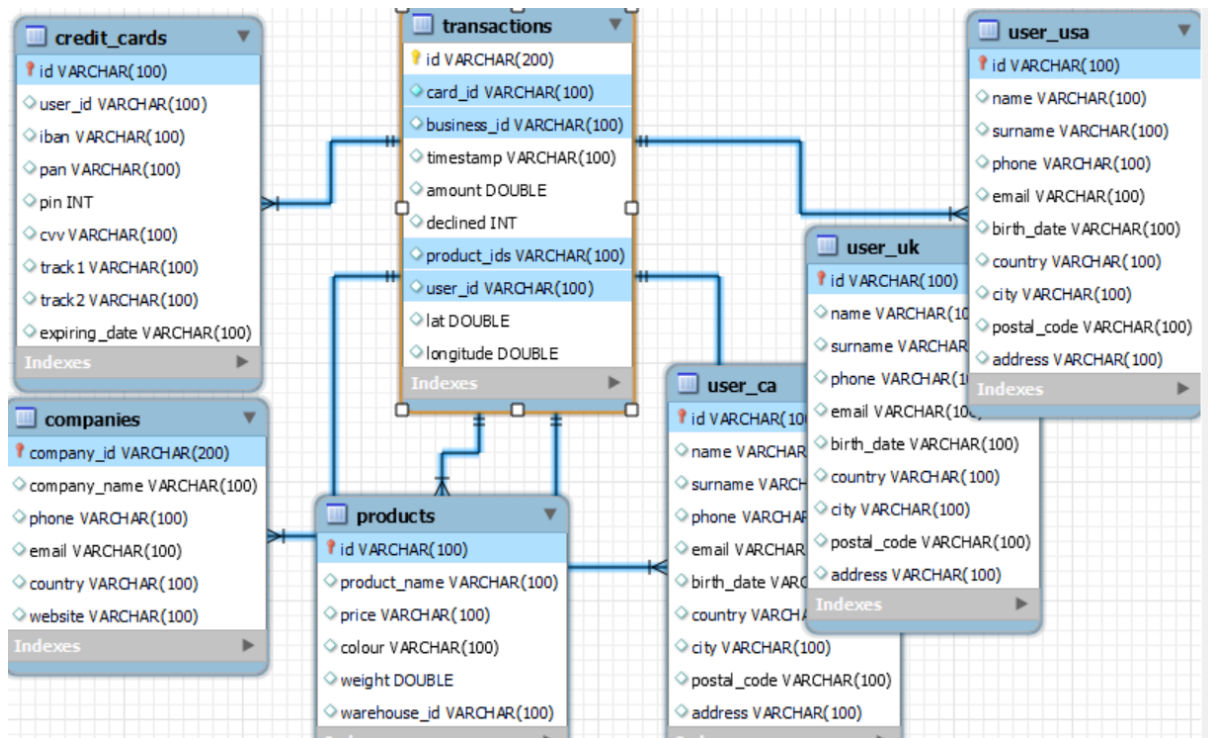
❖ añadido las foreign keys que me faltaban

```

1 • SET FOREIGN_KEY_CHECKS=0;
2
3 • alter table transactions add constraint fk_tr_pr foreign key (product_ids) references products(id);
4
5 • alter table transactions add constraint fk_us_ca foreign key (user_id) references user_ca(id);
6
7 • alter table transactions add constraint fk_us_us foreign key (user_id) references user_usa(id);
8
9 • alter table transactions add constraint fk_us_uk foreign key (user_id) references user_uk(id);
10

```

❖ muestro el modelo creado



❖ selecciono los usuarios con mas de 30 transacciones

The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```

3
4 • select distinct credit_cards.user_id 'usuarios amb més de 30 transaccions' from credit_cards
5   join transactions on transactions.card_id=credit_cards.id
6   where card_id in(
7     SELECT card_id FROM sprint4.transactions where declined=0 group by card_id having count(*) >30
8   );
9

```

The results pane shows a table with the following data:

usuarios amb més de 30 transaccions
272
267
92

Below the results pane, there is an 'Action Output' section showing the execution of the query:

#	Time	Action	Message
1	21:57:59	select distinct credit_cards.user_id 'usuarios amb més de 30 transaccions' from credit_...	3 row(s) returned
2	22:00:18	select distinct credit_cards.user_id 'usuarios amb més de 30 transaccions' from credit_...	3 row(s) returned

- código:  

```

select distinct credit_cards.user_id 'usuarios amb més de 30 transaccions' from credit_cards
join transactions on transactions.card_id=credit_cards.id
where card_id in(
    SELECT card_id FROM transactions where declined=0 group by card_id having
count(*) >30
);

```

- 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.

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: [I](#)

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

```
1 • ALTER TABLE transactions ADD activa varchar (10) default 'si' AFTER card_id;
2 • UPDATE transactions SET activa = CASE
3   WHEN card_id in (
4     select card_id from(
5       select * from(
6         select card_id,timestamp,declined,ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp desc) AS rn FROM transactions
7       ) x
8       where rn<4
9     ) y
10    group by card_id having sum(declined)>2
11  ) THEN REPLACE(activa, 'si', 'no')
12  else 'si'
13  END;
14 • select*from transactions;
```

Result Grid										
Filter Rows:										
Export:   Wrap Cell Content:   Fetch rows:										
id	card_id	activa	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
10881D1D-5B23-A76C-55EF-C568E49A05DD	CcU-2938	si	b-2222	2021-07-07 17:43:16	293.57	0	59	275	83.7839152128	-178.860353536
7DC26247-20EC-53FE-E555-B6C2E55CA9D5	CcU-2945	si	b-2226	2022-02-04 15:52:56	312.5	0	71, 41	275	58.9367181312	-76.8171099136
72997E96-DC2C-A4D7-7C24-66C302F8AE5A	CcU-2952	si	b-2230	2022-01-30 15:16:36	239.87	0	97, 41, 3	275	43.3584055296	-17.6579677184
AB069F53-965E-A2A8-CE06-CA8C4FD92501	CcU-2959	si	b-2234	2021-04-15 13:37:18	60.99	0	11, 13, 61, 29	275	1.6481916928	-158.0065729536
2F386AB6-147D-EB08-FE8D-9A4E2EA908D5	CcU-2966	si	b-2238	2021-10-18 06:12:03	33.81	0	47, 37, 11, 1	275	-43.4811227136	16.6025207808
5B0EEF86-B8A1-EFAA-SEE1-27E7DC8F54A4	CcU-2973	si	b-2242	2022-01-06 01:44:48	42.82	0	23, 19, 71	275	-64.1136375808	85.2490600448
2B928E1C-EC14-A760-0A75-871477649D6A	CcU-2980	si	b-2246	2021-08-10 08:14:49	383.73	0	59, 13, 23	275	-41.049559552	161.6848917504
063FBA79-99EC-66FB-29F7-25726D1764A5	CcU-2987	si	b-2250	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.222680576	-129.049879552
transactions 2 x										
Output:										
Action Output										
#	Time	Action	Message					Duration / Fetch		
1	22:56:49	ALTER TABLE transactions ADD activa varchar (10) default 'si' AFTER card_id	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0					0.016 sec		
2	22:56:49	UPDATE transactions SET activa = CASE WHEN card_id in ( select card_id from ( s...	0 row(s) affected Rows matched: 587 Changed: 0 Warnings: 0					0.313 sec		
3	22:56:49	select*from transactions LIMIT 0, 500	500 row(s) returned					0.000 sec / 0.000 s		

➤ código:

```
ALTER TABLE transactions ADD activa varchar (10) default 'si' AFTER card_id;
UPDATE transactions SET activa = CASE
  WHEN card_id in (
    select card_id from(
      select * from(
        select card_id,timestamp,declined,ROW_NUMBER() OVER (PARTITION BY card_id
ORDER BY timestamp desc) AS rn FROM transactions
      ) x
      where rn<4
    ) y
    group by card_id having sum(declined)>2
```

```
) THEN REPLACE(attiva, 'si', 'no')
else 'si'
END;
```

➤ explicación:

ALTER TABLE transactions ADD attiva varchar (10) default 'si' AFTER card\_id;  
-Creo la nueva columna donde aparecerà el estado de la tarjeta.

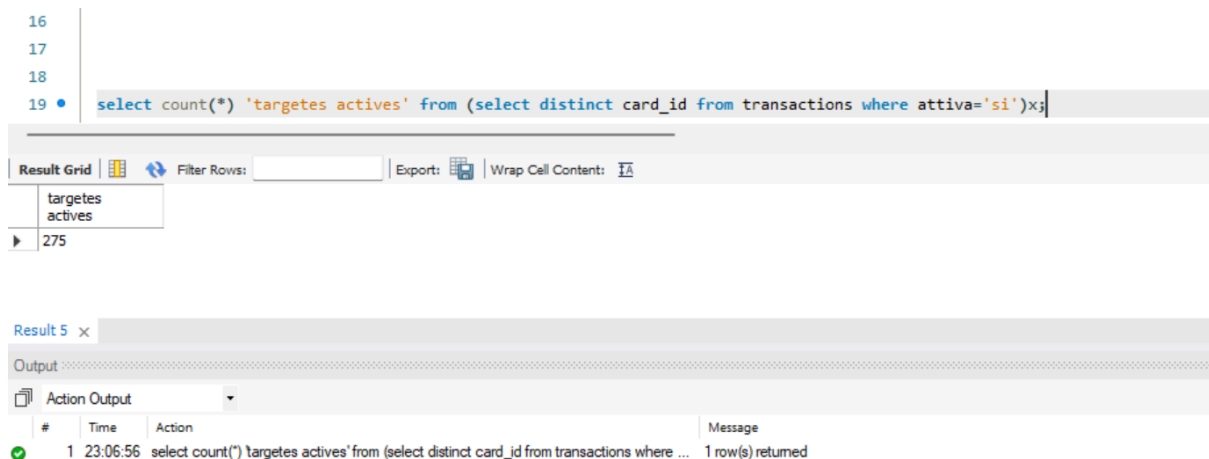
UPDATE transactions SET attiva = CASE WHEN card\_id in (..  
-Modifico la columna creada segun la condiciones requeridas.

select card\_id,timestamp,declined,ROW\_NUMBER() OVER (PARTITION BY card\_id ORDER  
-BY timestamp desc) AS rn FROM transactions  
Relleno una nueva columna temporal con 'Row Number' que asigna valores 1,2,3 a las  
ultimas tres transacciones agrupadas por 'card\_id'

```
where rn<4 ) y
group by card_id having sum(declined)>2
) THEN REPLACE(attiva, 'si', 'no')
else 'si'
```

-En caso que la suma de los declined de las ultimas tres transacciones (rn<4) sea mas de  
dos remplazo el valor 'si' con 'no'

### ❖ cuento las tarjetas activas



The screenshot shows a SQL IDE interface. At the top, a query is entered in the editor:

```
select count(*) 'targetes activas' from (select distinct card_id from transactions where attiva='si')x;
```

Below the editor, the 'Result Grid' is displayed, showing a single row with the value 275 under the column 'targetes activas'.

At the bottom, the 'Output' pane shows the execution details:

#	Time	Action	Message
1	23:06:56	select count(*) 'targetes activas' from (select distinct card_id from transactions where ...	1 row(s) returned

➤ código:

```
select count(*) 'targetes activas' from (select distinct card_id from transactions where  
attiva='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.

❖ muestro el resultado

```
1 • WITH RECURSIVE cte_count (n) AS (  
2     SELECT 1 UNION ALL SELECT n + 1  
3     FROM cte_count  
4     WHERE n < 1000)  
5  
6 • select idprod 'producte', count(*) 'nombre de vegades que s'ha venut' from(  
7     SELECT TRIM( BOTH FROM SUBSTRING_INDEX( SUBSTRING_INDEX(product_ids, ',', n) ,',',-1) )AS idprod  
8     FROM transactions  
9     JOIN cte_count cnt WHERE cnt.n <= LENGTH(product_ids) -LENGTH(REPLACE(product_ids,',','')) +1) x  
10 group by idprod  
11 order by idprod;
```



Result Grid

	producte	nombre de vegades que s'ha venut
▶	1	61
	11	48
	13	60
	17	61
...	...	...

Result 17 x

Output

Action Output

#	Time	Action	Message
✓ 1	10:58:08	WITH RECURSIVE cte_count (n) AS ( SELECT 1 UNION ALL SELECT n + 1 FR...	26 row(s) returned

➤ código:

```
WITH RECURSIVE cte_count (n) AS (  
    SELECT 1 UNION ALL SELECT n + 1  
    FROM cte_count  
    WHERE n < 1000)
```

```
select idprod 'producte', count(*) 'nombre de vegades que s'ha venut' 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  
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 < 1000)
```

-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 compra pase los 1000 productos.

-SELECT idprod 'producte', count(\*) 'nombre de veces que se ha vendido':

-Selecciono el ID del producto y cuento cuántas veces se ha vendido de los valores que voy a extraer que al final agruparé y ordenaré por el ID.

-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'.

