SPRINT 4

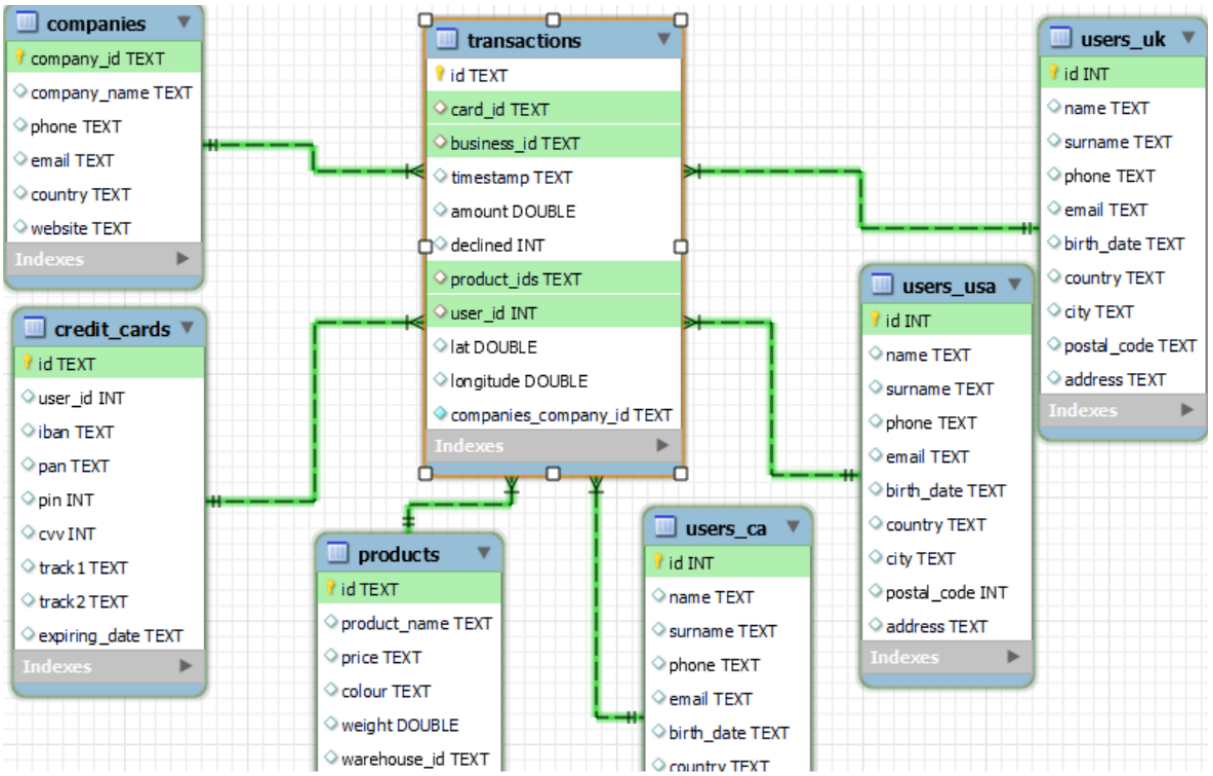
Edoardo Brega

★ NIVELL 1

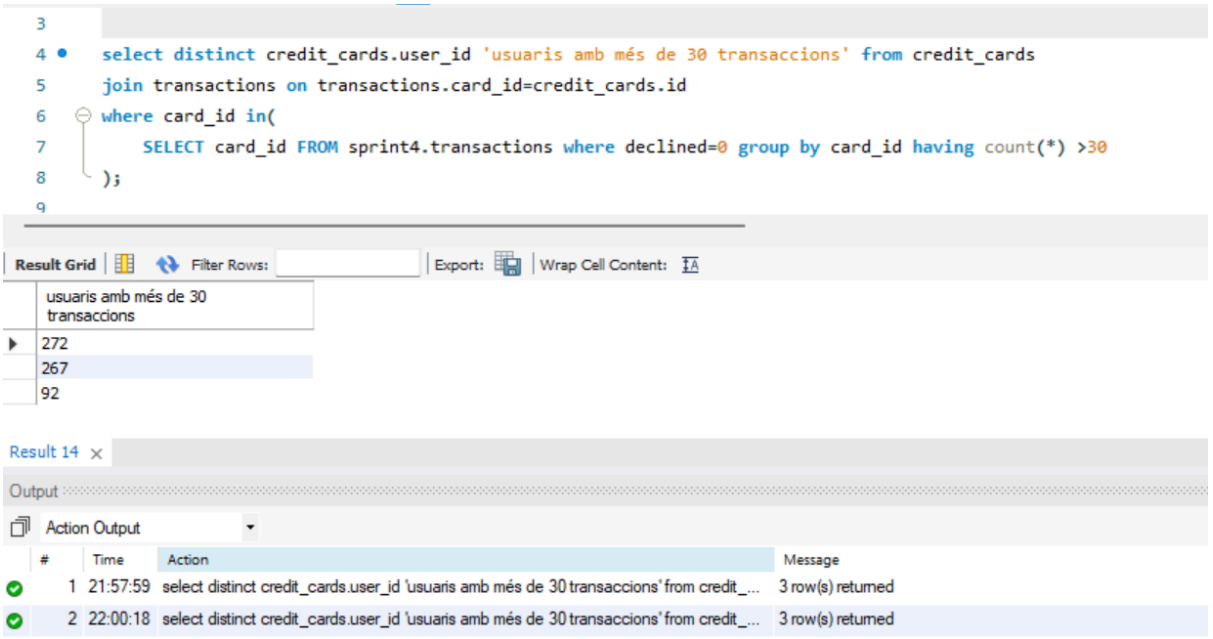
➔ exercici 1

- **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.

❖ **muestro el modelo creado**



❖ **selecciono los usuarios con mas de 30 transacciones**



```
3
4 •  select distinct credit_cards.user_id 'usuaris 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
```

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

usuaris amb més de 30 transaccions
272
267
92

Result 14

Output

Action Output

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

➢ código:

```
select distinct credit_cards.user_id 'usuaris 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.

➔ 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 compañia 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:

iban Donec Ltd	mitjana d´amount
PT87806228135092429456346	42.82

Result 16 | Result 17

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 compañia 'Donec Ltd'.

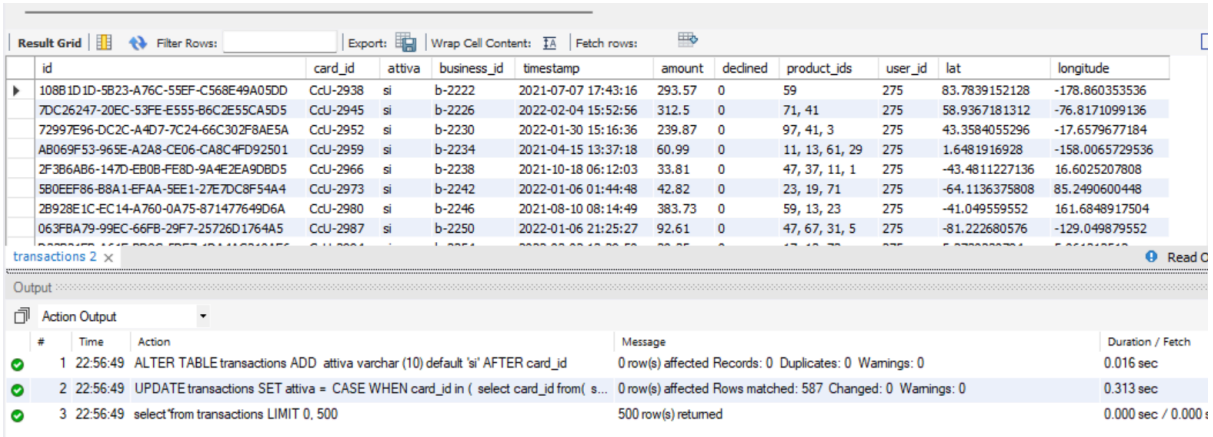
★ NIVELL 2

➔ 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 reflecione si las tarjetas estan activas**

```
1  ALTER TABLE transactions ADD  attiva varchar (10) default 'si' AFTER card_id;
2  UPDATE transactions SET attiva = 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(attiva, 'si', 'no')
12 else 'si'
13 END;
14 select*from transactions;
```



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

id	card_id	attiva	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
108B1D1D-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-B6C2E55CA5D5	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
2F3B6AB6-147D-EB0B-FE8D-9A4E2EA9DBD5	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-5EE1-27E7DC8F54A4	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 | Read C

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	22:56:49	ALTER TABLE transactions ADD attiva 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 attiva = 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

➢ codigo:

```
ALTER TABLE transactions ADD  attiva varchar (10) default 'si' AFTER card_id;
UPDATE transactions SET attiva = 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;
```

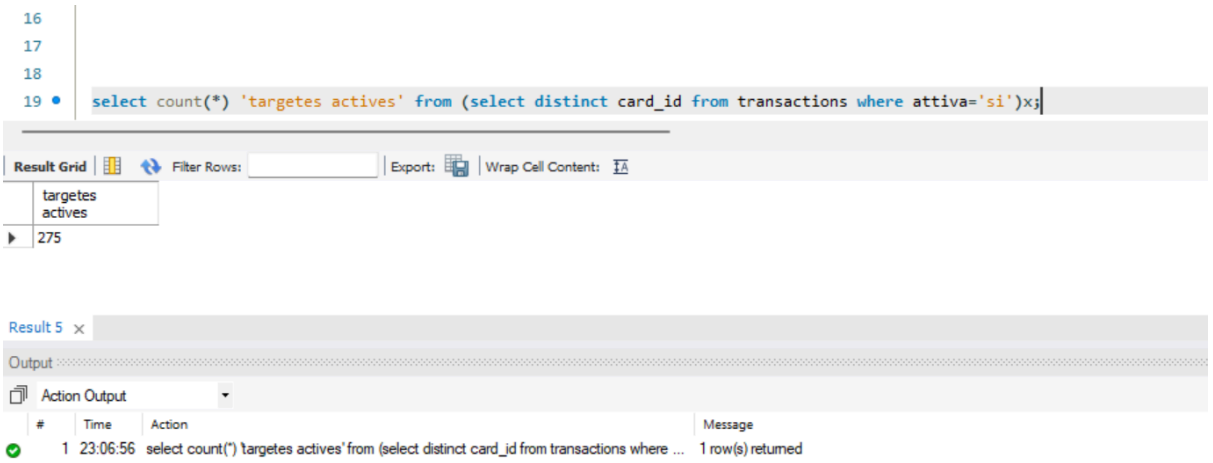
- explicación:
  ALTER TABLE transactions ADD attiva varchar (10) default 'si' AFTER card_id;
  -Creo la nueva columna donde aparecerà el estado de la tarjeta.

  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



- codigo:
  select count(*) ‘targetes actives’ 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 | Filter Rows: | Export: | Wrap Cell Content:

producte	nombre de vegades que s´ha venut
1	61
11	48
13	60
17	61

Result 17

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

➢ codigo:

```
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 codigo creo una tabla (Common Table Expression) con numeros sequenciales que necesitaré para relacionarla con la consulta siguiente.
Suponiendo que ninguna singola compra pase los 1000 productos.

-SELECT idprod ‘producte’, count(*) ‘nombre de vegades que s´ha venut’:
-Selecciono el ID del producto y cuenta 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 sin las comas nos da como resultado el numero de comas -1.
Con el +1 al final obtengo el numero de comas (que es igual al numero de productos-1) y me aseguro que ‘n’ no exceda el numero de productos en ‘poduct_ids’.