

Sprint 4 - Thais Alcaide

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:

Els arxius facilitats contenen la següent informació:

- Users CA: Noms i dades personals dels clients de Canada.
- Users UK: Noms i dades personals dels clients de UK.
- Users USA: Noms i dades personals dels clients de USA.

Aquests 3 arxius tenen informació en el mateix format de columnes i contingut, només canvia el país d'origen. Per respectar els arxius de càrrega (perquè potser el programa que els extreu només té capacitat per fer-ho per país i no ho pot treure tot d'una, o que interressi que ho faci així per algun motiu) inicialment crearé 3 tables, però després faré un UNION ALL per fer més senzill el diagrama final.

- Credit Cards: Informació de les targetes creades i a quin usuari estan assignades.
- Products: Id i dades de productes venuts, preu, warehouse on són..
- Companies: Dades de les empreses que venen aquests productes.
- Transactions: En aquesta taula tenim les dades de cada compra que s'ha fet. Inclou informació sobre els users, companies, credit cards i products, així que la podem utilitzar de taula de fets, al centre del model estrella, i al seu voltant estaràn la resta de taules com a dimensions.

Primer de tot he creat un nou database a mysql anomenat sprint 4, i utilitzo el USE per començar a treballar amb ell:

```
CREATE DATABASE IF NOT EXISTS sprint4;  
USE sprint4;
```

| | id | time | command | message | duration / result |
|---|-----|----------|---------------------------------------|-------------------|-------------------|
| ✓ | 626 | 12:21:26 | CREATE DATABASE IF NOT EXISTS sprint4 | 1 row(s) affected | 0.000 sec |
| ✓ | 627 | 12:21:26 | USE sprint4 | 0 row(s) affected | 0.000 sec |

Tot seguit creo 7 taules segons els arxius que hem revisat abans i pujo les dades dels arxius:

- Taula 1: Users_ca. L'id és el primary key. La primera columna és int perquè és numèrica, la resta són de tipus text variable, així que he assignat VARCHAR. La quantitat canvia segons el tipus d'informació i el que he observat que necessiten. La columna birth date no l'he posat DATE sinó VARCHAR perquè no té un format data acceptat per mysql (YYYY-MM-DD).

```
CREATE TABLE users_ca (  
  id int PRIMARY KEY,  
  name VARCHAR(20),  
  surname VARCHAR(20),  
  phone VARCHAR(30),  
  email VARCHAR(50),  
  birth_date VARCHAR(20),  
  country VARCHAR(20),  
  city VARCHAR(30),  
  postal_code VARCHAR(15),  
  address VARCHAR(100)  
);
```

23 12:03:38 CREATE TABLE users_ca (id int PRIMARY KEY, name VARCHAR(20), surname VARCHAR(20), phone VA... 0 row(s) affected

- Taula 2: Users_uk, igual que la taula anterior.

```
CREATE TABLE users_uk (  
  id int PRIMARY KEY,  
  name VARCHAR(20),  
  surname VARCHAR(20),  
  phone VARCHAR(30),  
  email VARCHAR(50),  
  birth_date VARCHAR(20),  
  country VARCHAR(20),  
  city VARCHAR(30),  
  postal_code VARCHAR(15),  
  address VARCHAR(100)  
);
```

18 11:58:33 CREATE TABLE users_uk (id int PRIMARY KEY, name VARCHAR(20), surname VARCHAR(20), phone VA... 0 row(s) affected

- Taula 3: Users_usa: Igual que les dues taules anteriors.

```
CREATE TABLE users_usa (  
    id int PRIMARY KEY,  
    name VARCHAR(20),  
    surname VARCHAR(20),  
    phone VARCHAR(30),  
    email VARCHAR(50),  
    birth_date VARCHAR(20),  
    country VARCHAR(20),  
    city VARCHAR(30),  
    postal_code VARCHAR(15),  
    address VARCHAR(100)  
);
```

21 12:01:43 CREATE TABLE users_usa (id int PRIMARY KEY, name VARCHAR(20), surname VARCHAR(20), phone V... 0 row(s) affected

Un cop creades aquestes 3 taules faig un UNION ALL per crear un esquema final de taules més senzill i ràpid d'utilitzar.

```
CREATE TABLE users AS  
SELECT * FROM users_usa  
UNION ALL  
SELECT * FROM users_uk  
UNION ALL  
SELECT * FROM users_ca;
```

2 10:00:51 CREATE TABLE users AS SELECT * FROM users_usa UNION ALL SELECT * FROM users_uk UNION ALL ... 275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

I assigno la primary key a Id:

```
ALTER TABLE users  
ADD PRIMARY KEY (id);
```

11 10:12:38 CREATE TABLE transactions (id VARCHAR(100) PRIMARY KEY, card_id VARCHAR(20), business_id VA... 0 row(s) affected

Ara sí, actualitzaré el format de la data de naixement amb la funció STR_TO_DATE i canviaré el format a la taula. La %b s'utilitza per anomenar els mesos abreviats (Dec), la %e per dies expressats sense un 0 davant (4, no 04) i la %Y pels anys amb 4 dígit (1981).

```
UPDATE users
SET birth_date = STR_TO_DATE(birth_date, '%b %e, %Y');
```

1 16:49:21 UPDATE users SET birth_date = STR_TO_DATE(birth_date, '%b %e, %Y')

275 row(s) affected Rows matched: 275 Changed: 275 Warnings: 0

```
ALTER TABLE users
MODIFY birth_date DATE;
```

2 16:49:28 ALTER TABLE users MODIFY birth_date DATE

275 row(s) affected Records: 275 Duplicates: 0 Warnings: 0

Ara ja ens queda un format DATE normatiu.

```
select *
from users;
```

| 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 | Mcconnell | (718) 257-2412 | integer.vitae.nibh@protonmail.org | 1992-08-23 | United States | Des Moines | 59464 | 903 Sit Ave |
| 3 | Ciaran | Harrison | (522) 598-1365 | interdum.feugiat@aol.org | 1998-04-29 | United States | Columbus | 56518 | 736-2063 Tellus St. |
| 4 | Howard | Stafford | 1-411-740-3269 | ornare.egestas@icloud.edu | 1989-02-18 | United States | Kailua | 77417 | Ap #545-2244 Erat. Rd. |
| 5 | Hayfa | Pierce | 1-554-541-2077 | et.malesuada.fames@hotmail.org | 1998-09-26 | United States | Sandy | 31564 | 341-2821 Ultrices Av. |
| 6 | Joel | Tyson | (718) 288-8020 | gravida.nunc.sed@yahoo.ca | 1989-10-15 | United States | Nashville | 96838 | 888-2799 Amet Street |
| 7 | Rafael | Jimenez | (817) 689-0478 | eget@outlook.ca | 1981-12-04 | United States | Hillsboro | 29874 | 8627 Malesuada Rd. |

- Taula 4, credit cards. L'id és la primary key. La majoria de columnes les he assignat com a VARCHAR, la longitud depenen de les necessitats de cada columna, menys user_id que és numèrica.

```
> CREATE TABLE credit_cards (
    id VARCHAR(20) PRIMARY KEY,
    user_id int,
    iban VARCHAR(50),
    pan VARCHAR(20),
    pin VARCHAR(4),
    cvv VARCHAR(4),
    track1 VARCHAR(100),
    track2 VARCHAR(100),
    expiring_date VARCHAR(10)
- );
```

34 12:58:24 CREATE TABLE credit_cards (id VARCHAR(20) PRIMARY KEY, user_id int, iban VARCHAR(50), pan VA... 0 row(s) affected

- Taula 5, companies. Company_id és la primary Key. Totes les columnes són de text així que he assignat VARCHAR, adaptant la longitud a les necessitats de cada tipus de data:

```
CREATE TABLE companies (  
    company_id VARCHAR(20) PRIMARY KEY,  
    company_name VARCHAR(50),  
    phone VARCHAR(20),  
    email VARCHAR(100),  
    country VARCHAR(20),  
    website VARCHAR(100)  
);
```

36 12:59:41 CREATE TABLE companies (company_id VARCHAR(20) PRIMARY KEY, company_name VARCHAR(50), ... 0 row(s) affected

- Taula 6: Transactions. Aquesta taula agrupa diferents tipus de dades. L'id és un text varchar que identifica cada transacció i es la primary key. A les que són foreign keys d'altres taules he afegit a quina taula i quina columna fan referència. Important que aquestes columnes tinguin el mateix format que els de la taula pare. L'amount és un decimal, declined és un tinyint perquè només té un dígit numeral, lat i longitud són floats.

```
CREATE TABLE transactions (  
    id VARCHAR(100) PRIMARY KEY,  
    card_id VARCHAR(20),  
    business_id VARCHAR(20),  
    timestamp TIMESTAMP,  
    amount decimal(10,2),  
    declined tinyint,  
    product_ids VARCHAR(20),  
    user_id int,  
    lat FLOAT,  
    longitude FLOAT,  
    FOREIGN KEY (card_id) REFERENCES credit_cards(id),  
    FOREIGN KEY (business_id) REFERENCES companies(company_id),  
    FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

11 10:12:38 CREATE TABLE transactions (id VARCHAR(100) PRIMARY KEY, card_id VARCHAR(20), business_id VA... 0 row(s) affected

Després carrego totes les dades des de l'arxiu csv utilitzant un load data, a l'infile poso la ruta de l'arxiu, especifico que es carregui a la taula users i faig algunes correccions com que els camps els ha de separar amb una coma o que la primera fila s'ha d'ignorar perquè té el nom de les columnes.

En el cas de les 3 primeres taules, les de users, he afegit el comando LINES TERMINATED BY perquè l'última columna d'address contenia símbols que interferien en la correcta càrrega de les dades i ajuntava algunes línies.

```
LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\users_ca.csv' INTO TABLE users_ca
FIELDS TERMINATED BY ','
ENCLOSED BY ''
LINES TERMINATED BY '\\r\\n'
IGNORE 1 ROWS;
```

25 12:06:08 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\users_ca.csv' INTO T... 75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0

27 12:14:17 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\users_uk.csv' INTO T... 50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Warnings: 0

28 12:14:37 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\users_usa.csv' INTO T... 150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0

A la resta de càrregues no ha fet falta aquesta línia de codi:

```
LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\credit_cards.csv' INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY ''
IGNORE 1 ROWS;
```

35 12:59:07 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\credit_cards.csv' INTO... 275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0

38 13:11:38 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\companies.csv' INTO ... 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Com a excepció la de transactions que necessitava la línia de Lines terminated i també canviar fields terminated by a ; sinó no es podia fer la càrrega. Això es pot veure en llegir l'arxiu csv amb un bloc de notes per veure bé les delimitacions de les columnes.

```
LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\transactions.csv' INTO TABLE transactions
FIELDS TERMINATED BY ';'
ENCLOSED BY ''
LINES TERMINATED BY '\\r\\n'
IGNORE 1 ROWS;
```

16 19:11:54 SELECT * FROM sprint4.transactions LIMIT 0, 1000

587 row(s) returned

Com a nota, al moment de carregar les dades del csv inicialment m'he trobat amb error de permís per carregar les dades a les taules que he creat:

636 13:11:02 LOAD DATA INFILE 'C:\Users\thais\OneDrive\Escritori\...' Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement

Primer he buscat el directori des d'on està permès pujar arxius a mysql:

```
SHOW VARIABLES LIKE 'secure_file_priv';
```

| Variable_name | Value |
|------------------|--|
| secure_file_priv | C:\ProgramData\MySQL\MySQL Server 8.0\Uploads\ |

Però després de passar els arxius a aquest directori i tornar a provar la càrrega no ha funcionat.

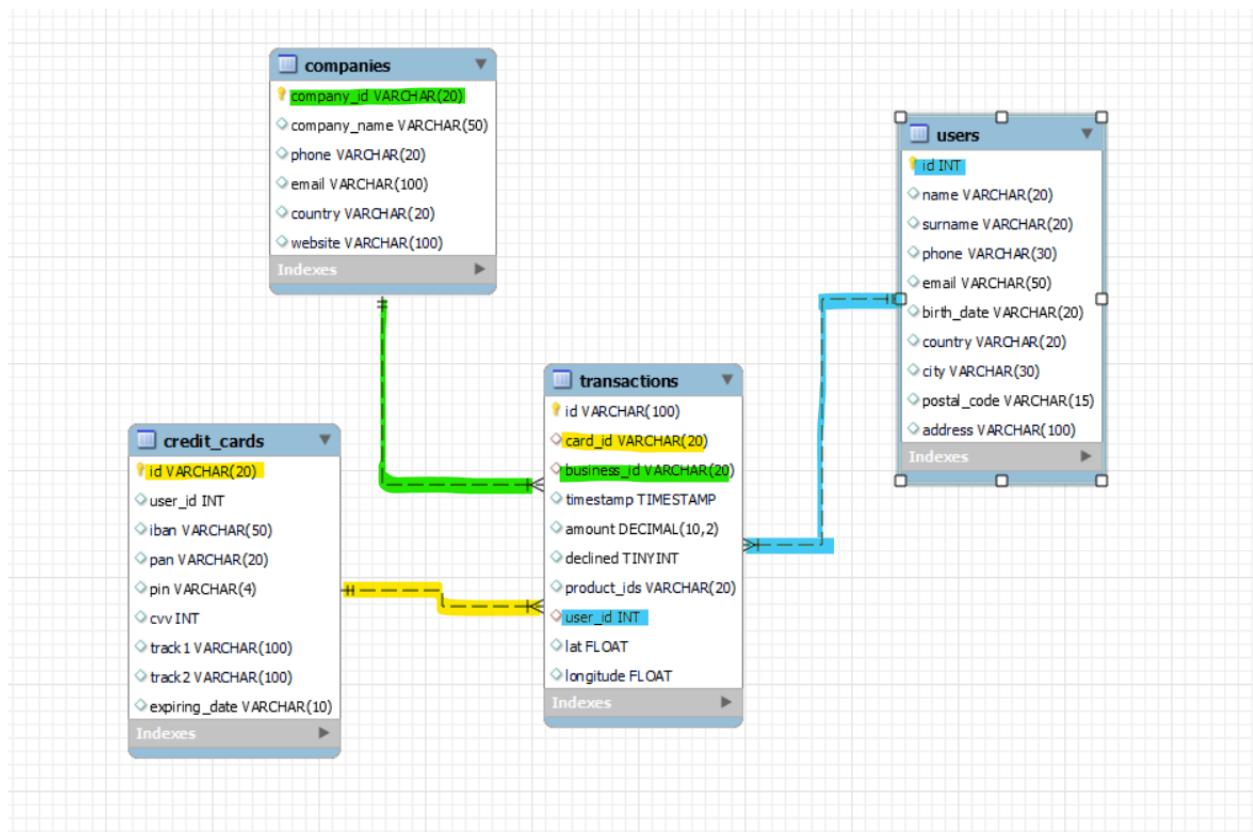
El següent intent ha estat modificant l'arxiu my.ini a: C:\ProgramData\MySQL\MySQL Server 8.0\my.ini. A l'apartat secure-file-priv="C:/ProgramData/MySQL/MySQL Server 8.0/Uploads", podries canviar el directori des d'on està permès pujar arxius o eliminar-lo amb NULL o "". Però el sistema no m'ha deixat guardar aquests canvis per falta de permisos.

Finalment, he trobat la forma de pujar aquests arxius des de la modalitat local. Primer he mirat si tenia habilitat el mode local amb show variables local infile, al veure que no l'he activat amb el set global = 1. A partir d'aquí no he tingut més problemes per pujar els arxius, només he hagut d'afegir el comando LOCAL a LOAD DATA.

```
show variables like "local_infile";  
set global local_infile = 1;
```

| | | | | |
|---|-----|----------|------------------------------------|-------------------|
| ✓ | 644 | 13:20:11 | show variables like "local_infile" | 1 row(s) returned |
| ✓ | 645 | 13:20:26 | set global local_infile = 1 | 0 row(s) affected |

Un cop carregada tota la informació puc extreure el diagrama d'aquest model d'estrella on queda la taula transaccions al centre com a taula de fets, on estan totes les mètriques, i la resta de taules al voltant com a dimensions que proporcionen informació més detallada de les dimensions. Les relacions són de 1 a N, com es pot veure a les línies de relació, on per exemple, a la taula users només apareixerà un cop cada id, però a la taula transactions pot aparèixer molts cops perquè aquest usuari ha comprat moltes vegades. He marcat les connexions amb colors perquè siguin més fàcils d'identificar. Les dades tenen molta granularitat, ja que tenen un gran nivell de detall a la taula transaccions, cada entrada representa una transacció individual.





- Exercici 1

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

A la taula de transaccions he fet un left join de la taula usuaris per tenir les dades que ens fan falta. Un cop tenim tota la informació recollida en una taula faig un count de l'id de transaccions per saber quantes transaccions hi ha en total i utilitzo GROUP BY per l'id de l'usuari per agrupar per usuari per saber quantes transaccions corresponen a cada persona. Després afegeixo un filtre HAVING (perquè el filtre és a una funció d'agregació i no podem utilitzar WHERE) que accepti només els resultats amb un count superior a 30.


```
SELECT u.id, u.name, u.surname, count(t.id) AS 'Num.transaccions'
FROM transactions AS t
LEFT JOIN users AS u ON t.user_id = u.id
GROUP BY u.id
HAVING count('Num.transaccions') > 30;
```

Result Grid





Filter Rows:

Export:



Wrap Cell Content:



| | id | name | surname | Num.transaccions |
|---|-----|--------|---------|------------------|
| ▶ | 92 | Lynn | Riddle | 39 |
| | 267 | Ocean | Nelson | 52 |
| | 272 | Hedwig | Gilbert | 76 |
| | 275 | Kenyon | Hartman | 48 |

✓ 26 10:40:32 SELECT u.id, u.name, u.surname, count(t.id) AS 'Num.transaccions' FROM transactions AS t LEFT JOIN user... 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.

Per fer aquesta consulta necessito dades de la taula transactions (l'amount), taula companies (poder filtrar pel nom de l'empresa, que és la dada que tenim), i la taula credit_cards (per poder filtrar per IBAN). Faig un left join de les 3 taules per tenir la informació recollida. Després faig un filtre WHERE on especifico el nom de la companyia que ens han consultat, i afegeixo al select una funció d'agregació avg de la columna amount per saber la mitjana i agrupo amb GROUP BY per IBAN:

```
SELECT c.company_name, cc.iban, round(avg(t.amount),2) AS 'Mitjana.import'
FROM transactions AS t
LEFT JOIN companies AS c ON t.business_id = c.company_id
LEFT JOIN credit_cards AS cc ON t.card_id = cc.id
WHERE c.company_name = 'Donec Ltd'
GROUP BY cc.iban;
```

| | company_name | iban | Mitjana.import |
|---|--------------|---------------------------|----------------|
| ▶ | Donec Ltd | PT87806228135092429456346 | 203.72 |

33 10:49:01 SELECT c.company_name, cc.iban, avg(t.amount) FROM transactions AS t LEFT JOIN companies AS c ON t... 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:

Primer he buscat la taula que em permeti visualitzar les 3 últimes transaccions de cada targeta i si van ser declinades o no. He utilitzat la funció row_number, fent la partició per card_id, que és la variable que volem estudiar i ordenant pel timestamp de forma descendent per tenir primer les transaccions més actuals. Després fem un filtre al where per delimitar el número de resultats per targeta a 3 o menys, si no han fet suficients transaccions.

```
SELECT card_id, timestamp, declined
FROM (
  SELECT row_number() OVER (PARTITION BY card_id ORDER BY timestamp desc
    ROWS UNBOUNDED PRECEDING) AS num_compres
    , card_id, timestamp, declined
  FROM transactions
) sub
WHERE num_compres <= 3;
```

| | card_id | timestamp | declined |
|---|----------|---------------------|----------|
| ▶ | CcU-2938 | 2022-03-12 09:23:10 | 0 |
| | CcU-2938 | 2022-03-09 20:53:59 | 0 |
| | CcU-2938 | 2022-02-24 11:01:42 | 0 |
| | CcU-2945 | 2022-02-04 15:52:56 | 0 |
| | CcU-2945 | 2021-06-15 00:26:29 | 1 |
| | CcU-2952 | 2022-01-30 15:16:36 | 0 |
| | CcU-2952 | 2021-05-06 05:33:39 | 1 |
| | CcU-2959 | 2022-03-16 14:01:36 | 0 |
| | CcU-2959 | 2022-03-04 02:48:32 | 0 |
| | CcU-2959 | 2022-02-28 00:10:50 | 0 |
| | CcU-2966 | 2021-10-18 06:12:03 | 0 |
| | CcU-2966 | 2021-06-02 06:19:00 | 1 |
| | CcU-2973 | 2022-01-06 01:11:40 | 0 |

Result 4 ×

Exercici 1

Quantes targetes estan actives?

Per fer aquesta consulta necessitarem una funció CASE on especifica que si la targeta ha estat rebutjada 3 vegades queda com inactiva, sinó segueix activa.

Al from he fet 2 subconsultes: una amb la taula que hem creat anteriorment que només té les 3 últimes transaccions de cada targeta.

```
FROM (  
    SELECT row_number() OVER (PARTITION BY card_id ORDER BY timestamp desc) AS num_compres,  
           card_id, timestamp, declined  
    FROM transactions  
    ) sub  
WHERE num_compres <= 3
```

I un altre amb la funció CASE, que utilitzarà la taula temporal anterior per revisar si les targetes han estat declinades 3 cops i establir si estan actives o desactivades.

```
SELECT card_id,  
       CASE  
           WHEN SUM(declined = 1) < 3 THEN 'Activada'  
           WHEN SUM(declined = 1) = 3 THEN 'Desactivada'  
       END as estat
```

Al principi volia utilitzar el CASE directament al SELECT principal, però em vaig trobar que això no em deixava després utilitzar la nova columna creada (estat) al WHERE que faré després per filtrar.

Seguim amb un GROUP BY card_id perquè ens interessa l'estat final de cada targeta i faig un left join amb la taula de credit_card per assegurar que tinc informació de totes les targetes, fins i tot les que no han fet transaccions.

```
GROUP BY card_id  
  
LEFT JOIN credit_cards AS cc ON sub2.card_id = cc.id
```

Fem un filtre al WHERE perquè la cerca només tingui en compte les targetes activades, que és el que ens demana l'enunciat. Utilitzem aquesta nova columna que he anomenat estat i he creat amb el CASE.

```
WHERE estat = 'Activada';
```

Finalment, al Select farem un count per tenir el nombre de targetes i afegirem l'estat per assegurar que són les que estan actives.

```
SELECT count(card_id) AS 'total targetes', estat
```

Així quedaria la query sencera:

```
• SELECT count(card_id) AS 'total targetes', estat
FROM (
  SELECT card_id,
    CASE
      WHEN SUM(declined = 1) < 3 THEN 'Activada'
      WHEN SUM(declined = 1) = 3 THEN 'Desactivada'
    END as estat
  FROM (
    SELECT row_number() OVER (PARTITION BY card_id ORDER BY timestamp desc) AS num_compres,
      card_id, timestamp, declined
    FROM transactions
  ) sub
  WHERE num_compres <= 3
  GROUP BY card_id
) sub2
LEFT JOIN credit_cards AS cc ON sub2.card_id = cc.id
WHERE estat = 'Activada';
```

| Result Grid | | Filter Rows: | Export: |
|----------------|----------|--------------|---------|
| total targetes | estat | | |
| 275 | Activada | | |

✓ 12 18:06:54 SELECT count(card_id), estat FROM (SELECT card_id, CASE WHEN SUM(declined = 1) < 3 THEN 'Activada' WHEN SUM(... 1 row(s) returned

En aquest cas, totes les targetes segueixen actives.

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:

- Taula 6, products. Id és la primary key, és un número pel qual he assignat format VARCHAR. La majoria de columnes són de text i les he assignat com a VARCHAR, menys weight que és numèrica amb decimals. Price la volia fer amb decimal també però en tenir les dades un símbol de \$, mysql no ho ha reconegut. He hagut de fer VARCHAR també.

```
CREATE TABLE products (  
    id VARCHAR(20) PRIMARY KEY,  
    product_name VARCHAR(100),  
    price VARCHAR(10),  
    colour VARCHAR(15),  
    weight DECIMAL(3,2),  
    warehouse_id VARCHAR(10)  
);
```

4 10:03:52 CREATE TABLE products (id VARCHAR(20) PRIMARY KEY, product_name VARCHAR(100), price VARCH... 0 row(s) affected

```
LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\products.csv' INTO TABLE products  
FIELDS TERMINATED BY ','  
ENCLOSED BY ''  
IGNORE 1 ROWS;
```

22 11:48:12 LOAD DATA LOCAL INFILE 'C:\\Users\\thais\\OneDrive\\Documentos\\MYSQL\\products.csv' INTO TA... 100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0

Un cop creada i carregada la taula amb els valors proporcionats, corregeixo la columna price. Primer elimino el símbol \$, el substitueixo amb REPLACE per un espai en blanc perquè no interfereixi.

```
UPDATE products
SET price = REPLACE(price, '$', '');
```

23 11:48:15 UPDATE products SET price = REPLACE(price, '\$', '') 100 row(s) affected Rows matched: 100 Changed: 100 Warnings: 0

Ara ja puc canviar el format de la columna price de VARCHAR(10) a DECIMAL.

```
ALTER TABLE products
MODIFY price DECIMAL(10,2);
```

Un cop creada la taula de products, per poder relacionar-la amb la taula transactions necessitem separar els product_ids de la taula transactions per fila, ja que actualment tenim varis product_ids dins d'un mateix registre.

```
select id, product_ids
from transactions;
```

| | id | product_ids |
|---|--------------------------------------|---------------|
| ▶ | 02C6201E-D90A-1859-B4EE-88D2986D3802 | 71, 1, 19 |
| | 0466A42E-47CF-8D24-FD01-C0B689713128 | 47, 97, 43 |
| | 063FBA79-99EC-66FB-29F7-25726D1764A5 | 47, 67, 31, 5 |
| | 0668296C-CDB9-A883-76BC-2E4C44F8C8AE | 89. 83. 79 |

Per solucionar-ho he utilitzat la funció SUBSTRING_INDEX. En aquesta funció especifiquem que product_ids es el registre del qual volem crear substrings, que els valors estan delimitats per comes, el número de vegades que ha de buscar el delimitador (Num_productes.r) i que volem que retorni els resultats a la dreta de la delimitació (-1) (si volem els de l'esquerra el número ha de ser en positiu). Això crearà una nova columna que he anomenat product_id_indiv.

```
SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', Num_productes.r), ',', -1) product_id_indiv
```

Al principi he utilitzat aquest codi tal qual, repetint tantes vegades com product_ids pot haver-hi al mateix registre (en aquest cas 4), però això donava lloc a una query molt llarga, perquè cada cop s'havia de fer un UNION ALL per unir aquests nous substrings a la taula:

```
from
((select substring_index(product_ids, ',', 1) as product_ids
  from transactions
 ) union all
 (select substring_index(substring_index(product_ids, ',', 2), ',', -1) as product_ids
  from transactions
  where product_ids like '%,%'
 ) union all
 (select substring_index(substring_index(product_ids, ',', 3), ',', -1) as product_ids
  from transactions
  where product_ids like '%,%,%'
 ) union all
 (select substring_index(substring_index(product_ids, ',', 4), ',', -1) as product_ids
  from transactions
  where product_ids like '%,%,%,%'
 )) sub
```

Per reduir la mida del codi he agrupat els UNION ALL com a Num_productes.r i he fet un inner join amb transactions amb una condició al JOIN que revisa que el nombre de registres generats sigui el mateix que productes a una fila.

Això s'aconsegueix fent una resta dels CHAR_LENGTH de cada fila de product_ids amb comes i sense. El resultat és el nombre de comes (delimitadors) per registre, que és el nombre de productes -1 (sempre hi ha una coma menys que productes).

Aquest codi s'adapta al nombre de product_ids de cada fila, sempre que no hi hagi més de 4.

```
(SELECT 1 r UNION ALL SELECT 2
 UNION ALL SELECT 3 UNION ALL SELECT 4) Num_productes INNER JOIN transactions
ON CHAR_LENGTH(product_ids)
-CHAR_LENGTH(REPLACE(product_ids, ',', ''))>=Num_productes.r-1;
```


Ens queda aquest codi, i al resultat podem veure que ara tenim una nova columna anomenada product_id_indiv on ha separat els productes de cada transacció. Això ens permetrà relacionar la taula products (id) amb transactions (product_id_indiv).

```
SELECT
  *,
  SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', Num_productes.r), ',', -1) product_id_indiv
FROM
  (SELECT 1 r UNION ALL SELECT 2
   UNION ALL SELECT 3 UNION ALL SELECT 4) Num_productes INNER JOIN transactions
ON CHAR_LENGTH(product_ids)
  -CHAR_LENGTH(REPLACE(product_ids, ',', ''))>=Num_productes.r-1;
```

| r | id | card_id | business_id | timestamp | amount | declined | product_ids | user_id | lat | longitude | product_id_indiv |
|---|--------------------------------------|----------|-------------|---------------------|--------|----------|---------------|---------|----------|-----------|------------------|
| 3 | 02C6201E-D90A-1859-B4EE-88D2986D3B02 | CcJ-2938 | b-2362 | 2021-08-28 23:42:24 | 466.92 | 0 | 71, 1, 19 | 92 | 81.9185 | -12.5276 | 19 |
| 2 | 02C6201E-D90A-1859-B4EE-88D2986D3B02 | CcJ-2938 | b-2362 | 2021-08-28 23:42:24 | 466.92 | 0 | 71, 1, 19 | 92 | 81.9185 | -12.5276 | 1 |
| 1 | 02C6201E-D90A-1859-B4EE-88D2986D3B02 | CcJ-2938 | b-2362 | 2021-08-28 23:42:24 | 466.92 | 0 | 71, 1, 19 | 92 | 81.9185 | -12.5276 | 71 |
| 3 | 0466A42E-47CF-8D24-FD01-C0B689713128 | CcJ-4219 | b-2302 | 2021-07-26 07:29:18 | 49.53 | 0 | 47, 97, 43 | 170 | -43.9695 | -117.525 | 43 |
| 2 | 0466A42E-47CF-8D24-FD01-C0B689713128 | CcJ-4219 | b-2302 | 2021-07-26 07:29:18 | 49.53 | 0 | 47, 97, 43 | 170 | -43.9695 | -117.525 | 97 |
| 1 | 0466A42E-47CF-8D24-FD01-C0B689713128 | CcJ-4219 | b-2302 | 2021-07-26 07:29:18 | 49.53 | 0 | 47, 97, 43 | 170 | -43.9695 | -117.525 | 47 |
| 4 | 063FBA79-99EC-66FB-29F7-25726D1764A5 | CcJ-2987 | b-2250 | 2022-01-06 21:25:27 | 92.61 | 0 | 47, 67, 31, 5 | 275 | -81.2227 | -129.05 | 5 |
| 3 | 063FBA79-99EC-66FB-29F7-25726D1764A5 | CcJ-2987 | b-2250 | 2022-01-06 21:25:27 | 92.61 | 0 | 47, 67, 31, 5 | 275 | -81.2227 | -129.05 | 31 |

13 10:53:49 SELECT *, SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', numbers.n), ',', -1) product_id_indiv... 1000 row(s) returned

Com a comentari, el problema de fer això és que estem duplicant els registres de la resta de columnes. Per tant, si volguéssim fer una nova consulta sobre els imports, usuaris o targetes, el resultat no seria acurat. Aquest és un dels problemes d'emmagatzemar diverses dades dins d'un mateix registre.

Exercici 1

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

Per poder resoldre aquesta consulta utilitzarem la funció SUBSTRING_INDEX que he comentat abans per crear una taula temporal anomenada 'sub' i que té les dades del product_id separades.

```
FROM (  
  SELECT  
    SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', Num_productes.r), ',', -1) as product_id_indiv,  
    transactions.product_ids, declined  
  FROM  
    (SELECT 1 r UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4) Num_productes  
  INNER JOIN transactions  
    ON CHAR_LENGTH(product_ids) - CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= Num_productes.r - 1  
) sub
```

Farem un join amb la taula products per poder obtenir el nom dels productes i un filtre WHERE de declined=0 per assegurar que les transaccions han estat acceptades. La consulta ens demana el nombre de cops que s'ha venut cada producte, i entenc que si el pagament ha sigut rebutjat el procés de venda no ha finalitzat correctament, l'empresa no rebrà aquest ingrés.

```
JOIN products p ON p.id = sub.product_id_indiv  
WHERE declined=0
```

Finalment, al SELECT ens interessa tenir l'id del producte, el seu nom i un count del nombre de vegades que surt el product id a la taula que hem creat per saber quantes vegades s'ha venut (total_vendes). Al tenir la funció d'agregació COUNT també hem d'afegir un GROUP BY a la query.

```
SELECT  
  sub.product_id_indiv, p.product_name,  
  COUNT(sub.product_ids) as total_vendes  
GROUP BY sub.product_id_indiv
```

Això seria la query completa i els resultats de venda de cada producte:

```
SELECT
  sub.product_id_indiv, p.product_name,
  COUNT(sub.product_ids) as total_vendes
FROM (
  SELECT
    SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', Num_productes.r), ',', -1) as product_id_indiv,
    transactions.product_ids, declined
  FROM
    (SELECT 1 r UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4) Num_productes
  INNER JOIN transactions
    ON CHAR_LENGTH(product_ids) - CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= Num_productes.r - 1
) sub
JOIN products p ON p.id = sub.product_id_indiv
WHERE declined=0
GROUP BY sub.product_id_indiv
ORDER BY sub.product_id_indiv;
```

| Result Grid | Filter Rows: | Export: | Wrap C |
|------------------|---------------------|--------------|--------|
| product_id_indiv | product_name | total_vendes | |
| 1 | Direwolf Stannis | 19 | |
| 11 | Karstark Dorne | 15 | |
| 13 | palpatine chewbacca | 17 | |
| 17 | skywalker ewok sith | 23 | |
| 19 | dooku solo | 18 | |

Result 6 ✕

15 11:39:15 SELECT sub.product_id_indiv, p.product_name, COUNT(sub.product_ids) as total_vendes FROM (SEL... 26 row(s) returned