Exercici 1

Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.

He descargado los archivos csv y he importado todos menos el de productos que ese nos lo piden más adelante. He creado las tablas de nuevo ya que me estaba dando bastantes errores a la hora de insertar las filas adaptandolas a las nuevas columnas del csv.

```
1 ● ⊖ CREATE TABLE credit_card (
        id VARCHAR(20) PRIMARY KEY,
          user_id INT,
          iban VARCHAR(34),
        pan VARCHAR(20),
5
        pin INT,
6
        CVV INT,
7
8
          track1 VARCHAR(255),
9
          track2 VARCHAR(255),
          expiring_date DATE
10
     ( );
11
12
13 ● ⊝ CREATE TABLE user (
14
       id INT PRIMARY KEY,
15
         name VARCHAR(100),
16
         surname VARCHAR(100),
17
        phone VARCHAR(50),
         email VARCHAR(150),
18
        birth_date VARCHAR(10),
19
        country VARCHAR(100),
        city VARCHAR(100),
        postal_code VARCHAR(20),
          address VARCHAR(255)
```

En el .sql esta el codigo completo de la creación de tablas , pero como es parecido a la Tarea 3 solo pongo esta parte en el informe.

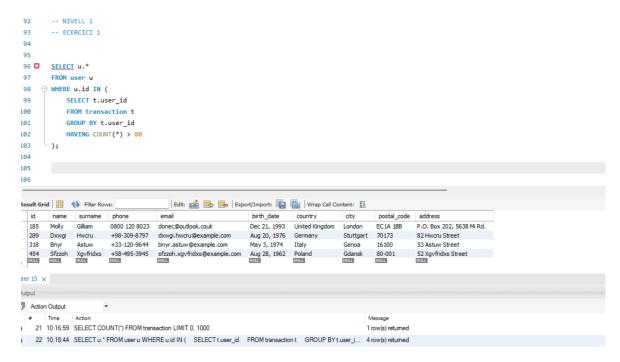
He insertado el csv mediante scripts tipo:

```
LOAD DATA LOCAL INFILE 'C:\\Users\\jordi\\Downloads\\transactions.csv' INTO TABLE transaction FIELDS TERMINATED BY ';' ENCLOSED BY '''' LINES TERMINATED BY '\n' IGNORE 1 LINES (id, card_id, business_id, `timestamp`, amount, declined, product_ids, user_id, lat, longitude);
```

Ya que el wizzard de inserción me iba excesivamente lento (unos 3s por fila).

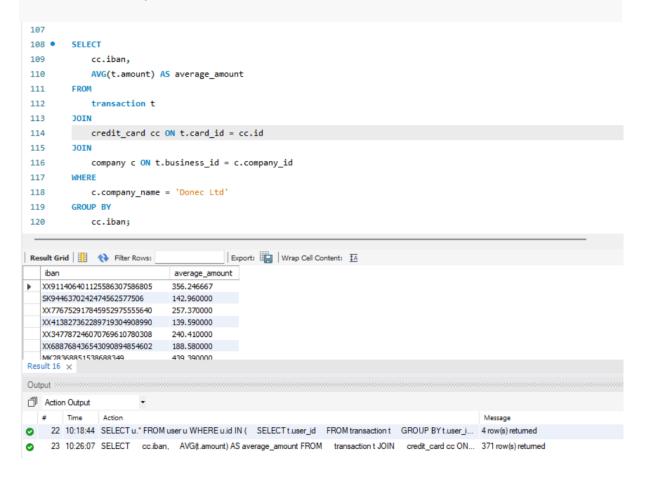
Una vez ya estaba todo insertado procedo ha hacer la query mediante un HAVING COUNT

para contabilizar la cantidad de usuarios con más de 80 transacciones.



Exercici 2

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



Se seleccionan solo las transacciones de "Donec Ltd", se enlazan con las tarjetas de crédito para obtener el IBAN de cada una, y se calcula la media de los importes por cada IBAN. Finalmente, se agrupan todos estos resultados por IBAN para ver claramente cuál es el promedio de gastos por cada tarjeta.

Nivell 2

Exercici 1

Quantes targetes estan actives?

Primero creamos una nueva tabla para guardar el card_id (identificador único) y un booleano para apuntar si las 3 últimas transacciones han sido declinadas o no

Ahora que hemos creado la tabla, lo que he hecho ha sido meter todas las filas de credit card a la nuevo tabla y poniendo por default last three declined=False.

```
-- METEMOS TODAS LAS TARGETAS

132

133 • INSERT INTO credit_card_status (card_id, last_three_declined)

134 SELECT DISTINCT id, FALSE

135 FROM credit_card

136 ON DUPLICATE KEY UPDATE last_three_declined = FALSE;
```

Una vez tenemos todos los registros de tarjetas de crédito con este Script vemos si las 3 últimas transacciones fueron aprobada y en caso positivo modificamos la fila en cuestión para poner a TRUE last_three_declined

```
-- MODIFICAMOS EL BOOLEANO SEGUN LAS 3 ULTIMAS TRANSACCIONES:
138
139
140 • ⊝ WITH ultimas_transacciones AS (
141
           SELECT
142
                t.card id,
143
                t.declined,
                ROW_NUMBER() OVER (PARTITION BY t.card_id ORDER BY t.timestamp DESC) AS rn
145
            FROM transaction t
      ١,,
147 ⊝ filtradas AS (
            SELECT card_id
149
            FROM ultimas_transacciones
           GROUP BY card id
151
           HAVING SUM(declined) = 3
      (
153
154
        UPDATE credit_card_status
       JOIN filtradas USING (card id)
155
       SET last_three_declined = TRUE;
157
Output :::
Action Output
               Action
                                                                                                 Message
31 11:12:04 INSERT INTO credit_card_status (card_id, last_three_declined) SELECT DISTINCT id, FALSE FROM credit_c... 5005 row(s) affected Records: 5000 Duplicates: 5 Warnings: 0
2 32 11:13:00 WITH ultimas_transacciones AS ( SELECT t.card_id, t.declined, ROW_NUMBER() OVER (P... 5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0
```

Primero, se crea una especie de tabla temporal llamada "ultimas_transacciones" donde se seleccionan todas las transacciones y se asigna un número a cada una por tarjeta, ordenando desde la más reciente a la más antigua. Esto sirve para identificar las tres últimas transacciones de cada tarjeta. Después, se filtran estas transacciones para quedarse solo con las tres primeras de cada tarjeta y se agrupan por tarjeta, calculando la suma de los valores de "declined" (que indica si la transacción fue rechazada). Si la suma es tres, quiere decir que las tres últimas transacciones fueron todas declinadas.

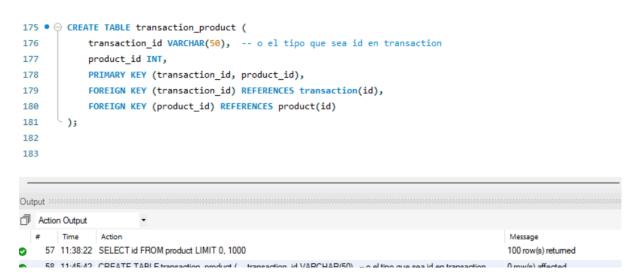
Nivell 3

Exercici 1

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

```
163 • ○ CREATE TABLE product (
164
             id INT PRIMARY KEY,
165
             product_name VARCHAR(255),
166
             price DECIMAL(10, 2),
             colour VARCHAR(20),
167
            weight DECIMAL(10, 2),
168
             warehouse id VARCHAR(20)
169
170
        );
        ALTER TABLE product MODIFY price VARCHAR(20);
171 •
```

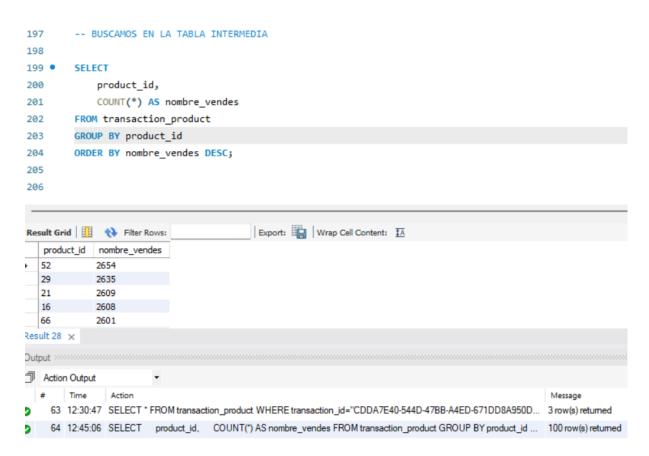
Como vemos que en transactions se referencian en product_ids a varios elementos a la vez, no podemos simplemente enlazarla con products así como sí nada. Para ello tenemos que crear una tabla intermedia donde se relacione de forma individual el transaction_id y el product_id.



Ahora tenemos que rellenar esta tabla con la información que hay en transactions para normalizar los elementos de product_ids y poder relacionarlos correctamente con la tabla products. Como puede haber diferente número de elementos en product_ids, tenemos que hacer un algoritmo que tenga esto en cuenta y vaya añadiendo de forma continua cada elemento de product_ids en una linea en la tabla intermedia junto con el transaction_id.

```
-- RELLENAMOS LA TABLA INTERMEDIA A PARTIR DE TRANSACTION
185
        INSERT INTO transaction product (transaction id, product id)
186
        SELECT
187
            t.id,
188
            CAST(TRIM(j.value) AS UNSIGNED) AS product_id
189
        FROM
190
            transaction t,
191
            JSON TABLE(
192
                CONCAT('[', REPLACE(t.product_ids, ',', ','), ']'),
193
                 "$[*]" COLUMNS(value VARCHAR(10) PATH "$")
194
195
            ) j;
```

Una vez hecho esto ya tenemos la tabla intermedia rellenada y entrelazada con las tablas transaction y product, por lo tanto ya podemos hacer la query que nos decia el enunciado.



Hacemos la búsqueda de forma sencilla ya que simplemente es agrupar por product_id y contarlos. Sabemos que es correcto debido a que el csv de products tenia 100 elementos y justo nos ha devuelto 100 filas como se puede ver abajo a la derecha de la captura.