

- NIVEL 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:

Al intentar crear las claves externas, hubo un error en la relación entre la tabla user_usa y la tabla transaccion, había valores de user_id de la tabla transaccion que faltaban en id en la tabla user_usa.

Se han agregado los datos y se han creado las claves externas.

Estructura de las tablas.

```
CREATE TABLE IF NOT EXISTS company (  
    id VARCHAR(20) PRIMARY KEY,  
    company_name VARCHAR(100),  
    phone VARCHAR(15),  
    email VARCHAR(100),  
    country VARCHAR(100),  
    website VARCHAR(255)  
);
```

```
CREATE TABLE IF NOT EXISTS user (  
    id VARCHAR (10) PRIMARY KEY,  
    name VARCHAR(100),  
    surname VARCHAR(100),  
    phone VARCHAR(150),  
    email VARCHAR(150),  
    birth_date VARCHAR(100),  
    country VARCHAR(50),  
    city VARCHAR(50),  
    postal_code VARCHAR(100),  
    address VARCHAR(255)  
);
```

```
CREATE TABLE IF NOT EXISTS credit_card (  
    id VARCHAR (15) PRIMARY KEY,  
    user_id VARCHAR (10),  
    iban VARCHAR (100),  
    pan VARCHAR (100),  
    pin VARCHAR (4),  
    cvv VARCHAR (3),  
    track_1 VARCHAR (255),  
    track_2 VARCHAR (255),  
    expiring_date VARCHAR (10)  
);
```

```
CREATE TABLE IF NOT EXISTS transaction (  
  id VARCHAR (255) PRIMARY KEY,  
  card_id VARCHAR (15),  
  company_id VARCHAR (20),  
  timestamp TIMESTAMP,  
  amount DECIMAL (10, 2),  
  declined BOOLEAN,  
  product_ids VARCHAR (20),  
  user_id VARCHAR (10),  
  lat FLOAT,  
  longitude FLOAT  
);
```

Agregando las informaciones en las tablas:

```
SET GLOBAL local_infile = 1;
```

```
LOAD DATA LOCAL INFILE
```

```
"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/comp  
anies.csv"
```

```
  INTO TABLE company  
  FIELDS TERMINATED BY ','  
  ENCLOSED BY '"'  
  LINES TERMINATED BY '\r\n'  
  IGNORE 1 LINES;
```

```
LOAD DATA LOCAL INFILE
```

```
"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/credit  
_cards.csv"
```

```
  INTO TABLE credit_card  
  FIELDS TERMINATED BY ','  
  ENCLOSED BY '"'  
  LINES TERMINATED BY '\n'  
  IGNORE 1 LINES;
```

LOAD DATA LOCAL INFILE

"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/transactions.csv"

INTO TABLE transaction

FIELDS TERMINATED BY ';' -- Cambiado para semicolon

#ENCLOSED BY ''

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE

"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/users_usa.csv"

INTO TABLE user -- Usuarios de USA

FIELDS TERMINATED BY ',' -- Indica el separador de campo

ENCLOSED BY '' -- Cómo se delimitan las cadenas de texto

LINES TERMINATED BY '\r\n' -- Carriage Return (CR) i Line Feed (LF)

IGNORE 1 LINES; -- Ignora la primera línea que contiene el nombre de la columna.

LOAD DATA LOCAL INFILE

"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/users_ca.csv"

INTO TABLE user

FIELDS TERMINATED BY ','

ENCLOSED BY ''

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

LOAD DATA LOCAL INFILE

**"C:/Users/Victor/Desktop/Especializacion_Datos/Sprint_4/users
_uk.csv"**

INTO TABLE user

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

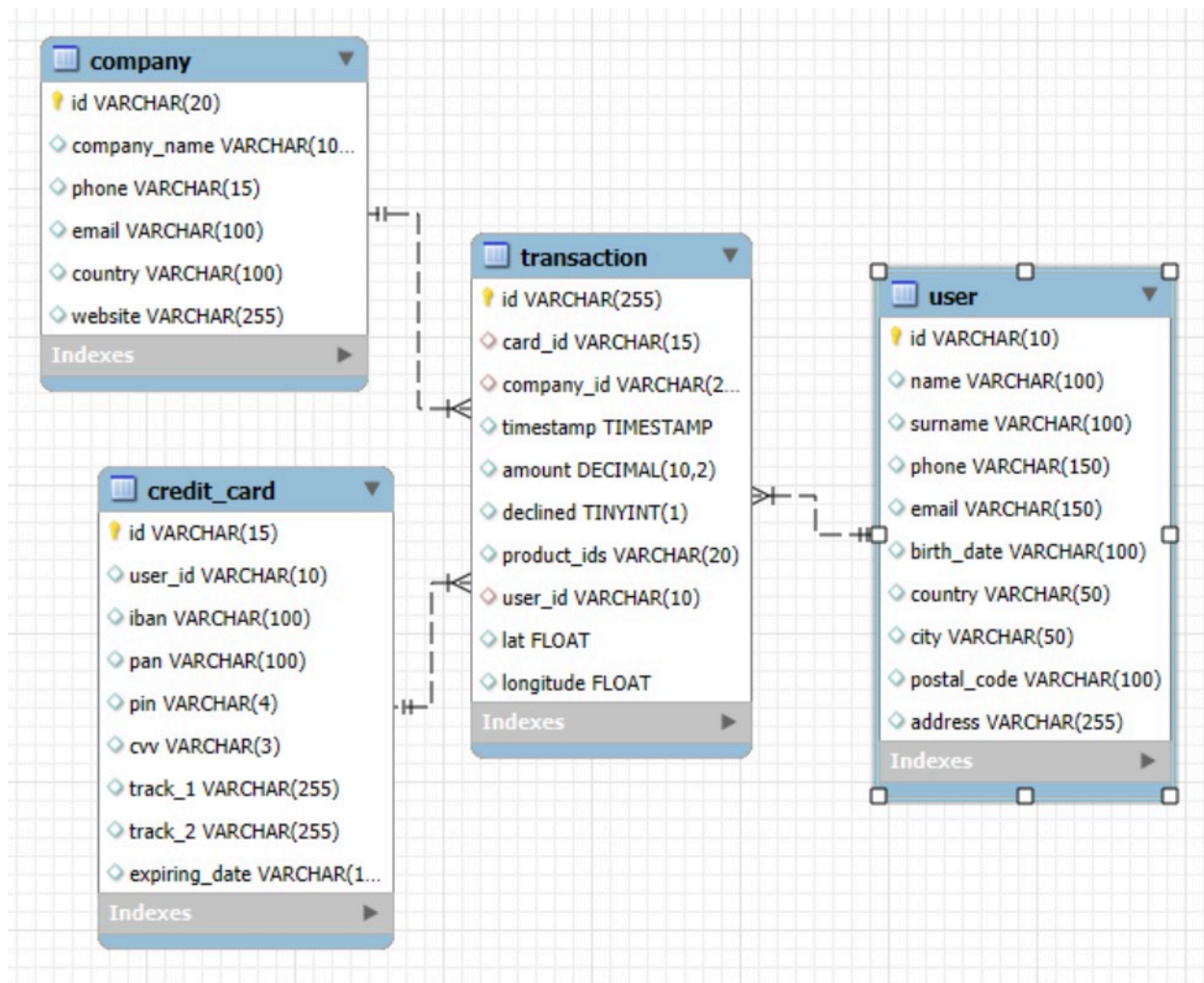
creando las foreign keys:

ALTER TABLE transaction

**ADD CONSTRAINT fk_company_id FOREIGN KEY (company_id)
REFERENCES company(id),**

**ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id)
REFERENCES user_usa(id),**

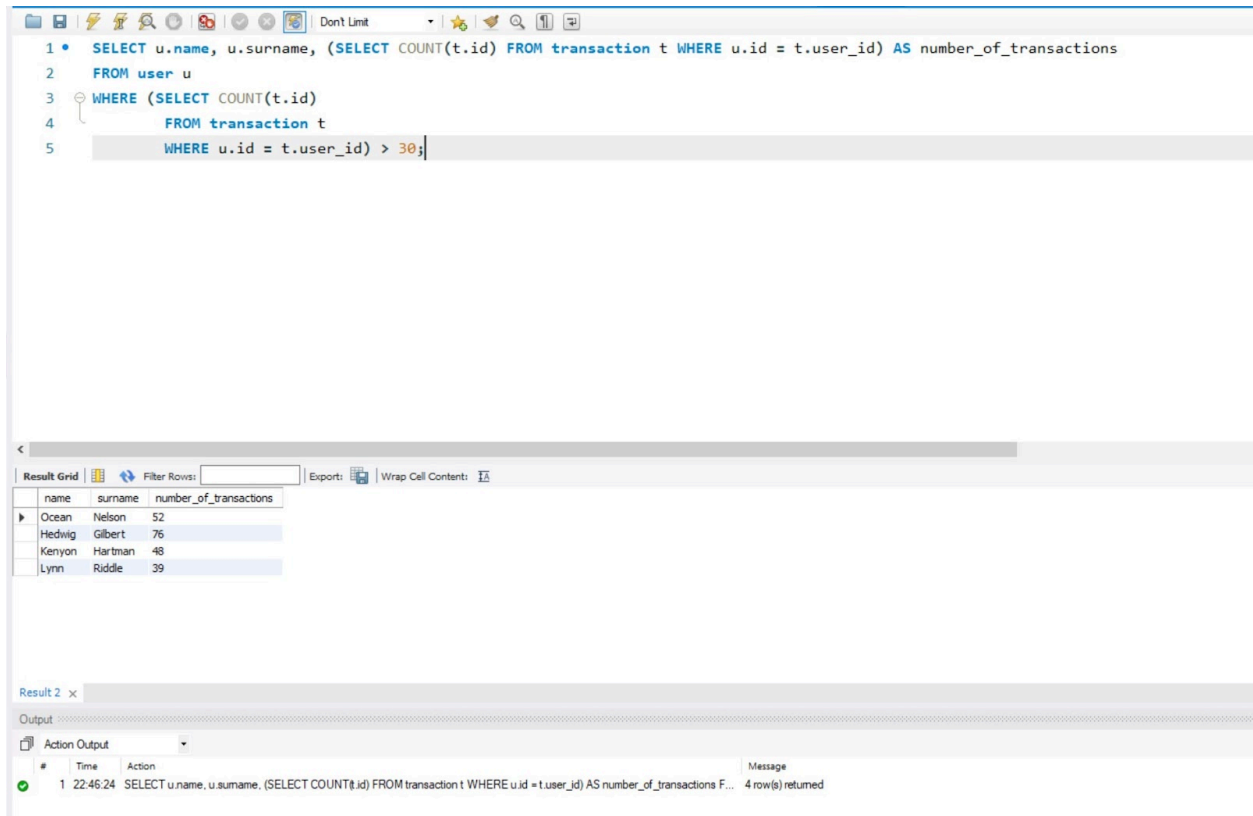
**ADD CONSTRAINT fk_card_id FOREIGN KEY (card_id)
REFERENCES credit_card(id);**



Se creó la base de datos **sprint_4**, dentro de ella se crearon 4 tablas: **transaction**(tabla de hechos), **company**(tabla de dimensiones), **user**(tabla de dimensiones) y **credit_card**(tabla de dimensiones).

EJERCICIO 1

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



The screenshot shows a SQL IDE interface. The top pane contains a SQL query that selects user names and surnames, along with a subquery counting transactions for each user. The bottom pane displays the results of the query in a table format.

```
1 • SELECT u.name, u.surname, (SELECT COUNT(t.id) FROM transaction t WHERE u.id = t.user_id) AS number_of_transactions
2 FROM user u
3 WHERE (SELECT COUNT(t.id)
4 FROM transaction t
5 WHERE u.id = t.user_id) > 30;
```

Result Grid

	name	surname	number_of_transactions
▶	Ocean	Nelson	52
	Hedwig	Gilbert	76
	Kenyon	Hartman	48
	Lynn	Riddle	39

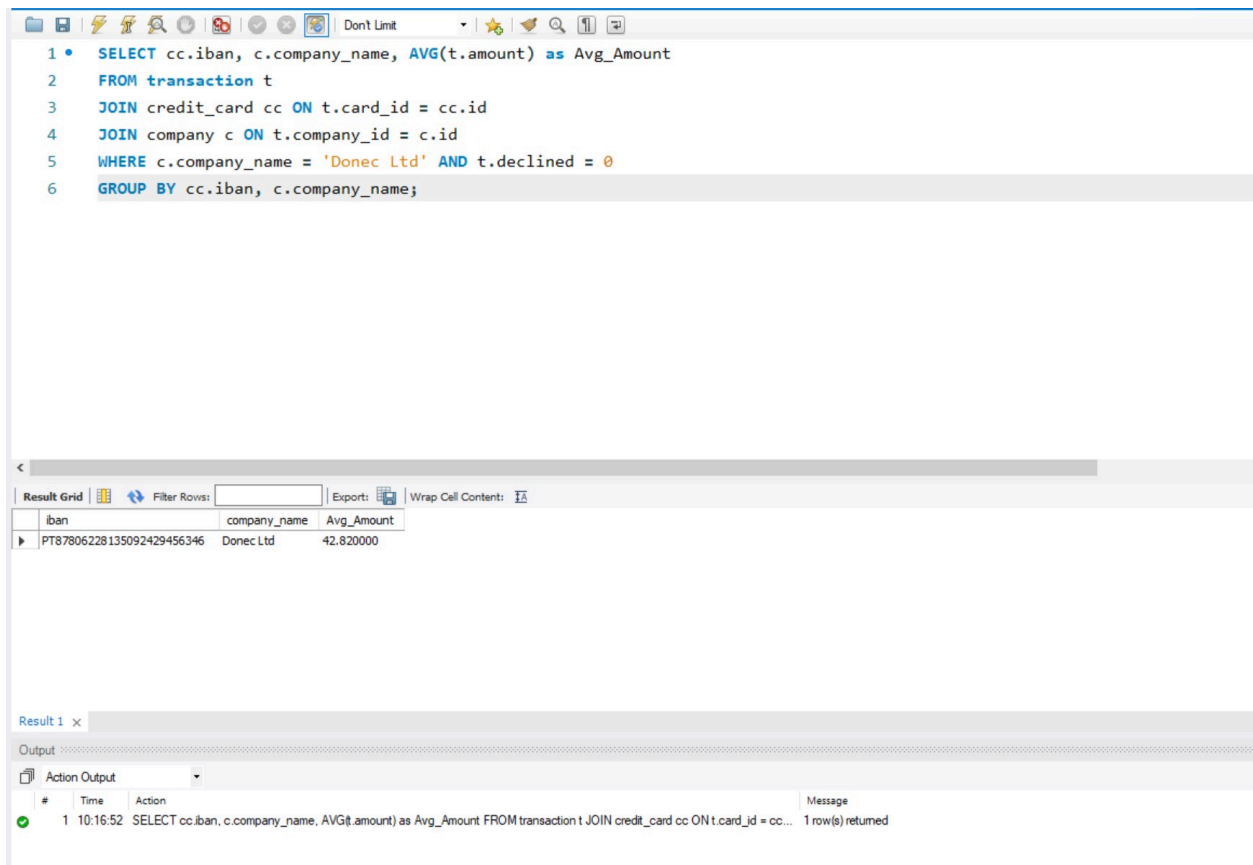
Result 2 x

Output

#	Time	Action	Message
1	22:45:24	SELECT u.name, u.surname, (SELECT COUNT(t.id) FROM transaction t WHERE u.id = t.user_id) AS number_of_transactions F...	4 row(s) returned

EJERCICIO 2

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



The screenshot shows a SQL IDE interface with a query editor at the top and a results pane at the bottom. The query editor contains a SQL query that selects the average amount of transactions for a specific company and card, joining the transaction, credit card, and company tables.

```
1 • SELECT cc.iban, c.company_name, AVG(t.amount) as Avg_Amount
2 FROM transaction t
3 JOIN credit_card cc ON t.card_id = cc.id
4 JOIN company c ON t.company_id = c.id
5 WHERE c.company_name = 'Donec Ltd' AND t.declined = 0
6 GROUP BY cc.iban, c.company_name;
```

The results pane displays a single row of data in a table format:

iban	company_name	Avg_Amount
PT87806228135092429456346	Donec Ltd	42.820000

Below the table, the output pane shows the execution details of the query, including the time taken and the message indicating that 1 row(s) were returned.

Result 1 x

Output

Action Output

#	Time	Action	Message
1	10:16:52	SELECT cc.iban, c.company_name, AVG(t.amount) as Avg_Amount FROM transaction t JOIN credit_card cc ON t.card_id = cc...	1 row(s) returned

- NIVEL 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:

Creando la estructura de la tabla:

```
CREATE TABLE IF NOT EXISTS card_status (  
    card_id VARCHAR (15) PRIMARY KEY,  
    status VARCHAR (15)  
);
```

Creando los parámetros de las columnas:

```
INSERT INTO card_status (card_id, status)  
SELECT card_id, CASE WHEN SUM(declined) = 3 THEN 'Blocked'  
ELSE 'Active' END as Status  
FROM (  
    SELECT card_id, declined, ROW_NUMBER()  
OVER(PARTITION BY card_id ORDER BY timestamp DESC) as rn  
    FROM transaction  
) as sub  
WHERE rn <= 3  
GROUP BY card_id;
```

Creando la coneccion con la tabla transaction

```
ALTER TABLE transaction
```

**ADD CONSTRAINT fk_card_status FOREIGN KEY
(card_id) REFERENCES card_status(card_id);**

EJERCICIO 1

Quantes targetes estan actives?

The screenshot shows a SQL IDE interface. The top pane contains the following SQL query:

```
1 • SELECT COUNT(*) AS active_cards
2 FROM card_status
3 WHERE status = 'Active';
```

The bottom pane displays the results of the query. The 'Result Grid' shows a single row with the value 275 for the column 'active_cards'.

active_cards
275

The 'Output' pane shows the execution details:

#	Time	Action	Message	Duration
1	10:35:48	SELECT COUNT(*) AS active_cards FROM card_status WHERE status = 'Active'	1 row(s) returned	0.016 s

- NIVEL 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:

Estructura de la taula:

```
CREATE TABLE IF NOT EXISTS product (  
    id VARCHAR (10) PRIMARY KEY,  
    product_name VARCHAR (100),  
    price VARCHAR (100),  
    color VARCHAR (100),  
    weight VARCHAR (10),  
    warehouse_id VARCHAR (255)  
);
```

Al conectar la tabla 'product' con la tabla 'transaction' hubo un problema.

La columna 'product_ids' tenía varios valores por campos, separados por comas, por lo que no era posible utilizarla.

Hay que crear una tabla de unión entre 'transaction' y 'product' y la llamé 'transaction_product'.

Lo que hice primero, pero estuvo mal, fue crear 1 tabla llamada transaction_product_1, en esta tabla coloqué solo las columnas product_ids y transacciones del archivo de transactions, seleccioné separarlas con una coma y separé cada ID de producto en una

columna diferente, luego creé consultas para organizar los datos hasta obtener el resultado correcto y luego exporté estos datos a un archivo CSV para colocar en la tabla final transaction_product, pero no se puede hacer de esta manera.

Al final, se creó una query para separar los datos de product_ids en una sola columna con los valores de transaction_id correspondientes y agregarlos en la tabla transaction_product, Tener en cuenta que también estoy cogiendo los valores NULL.

Tabla de unión:

```
CREATE TABLE IF NOT EXISTS transaction_product (  
    transaction_id VARCHAR (255),  
    product_id VARCHAR (10),  
    PRIMARY KEY (transaction_id, product_id)  
);
```

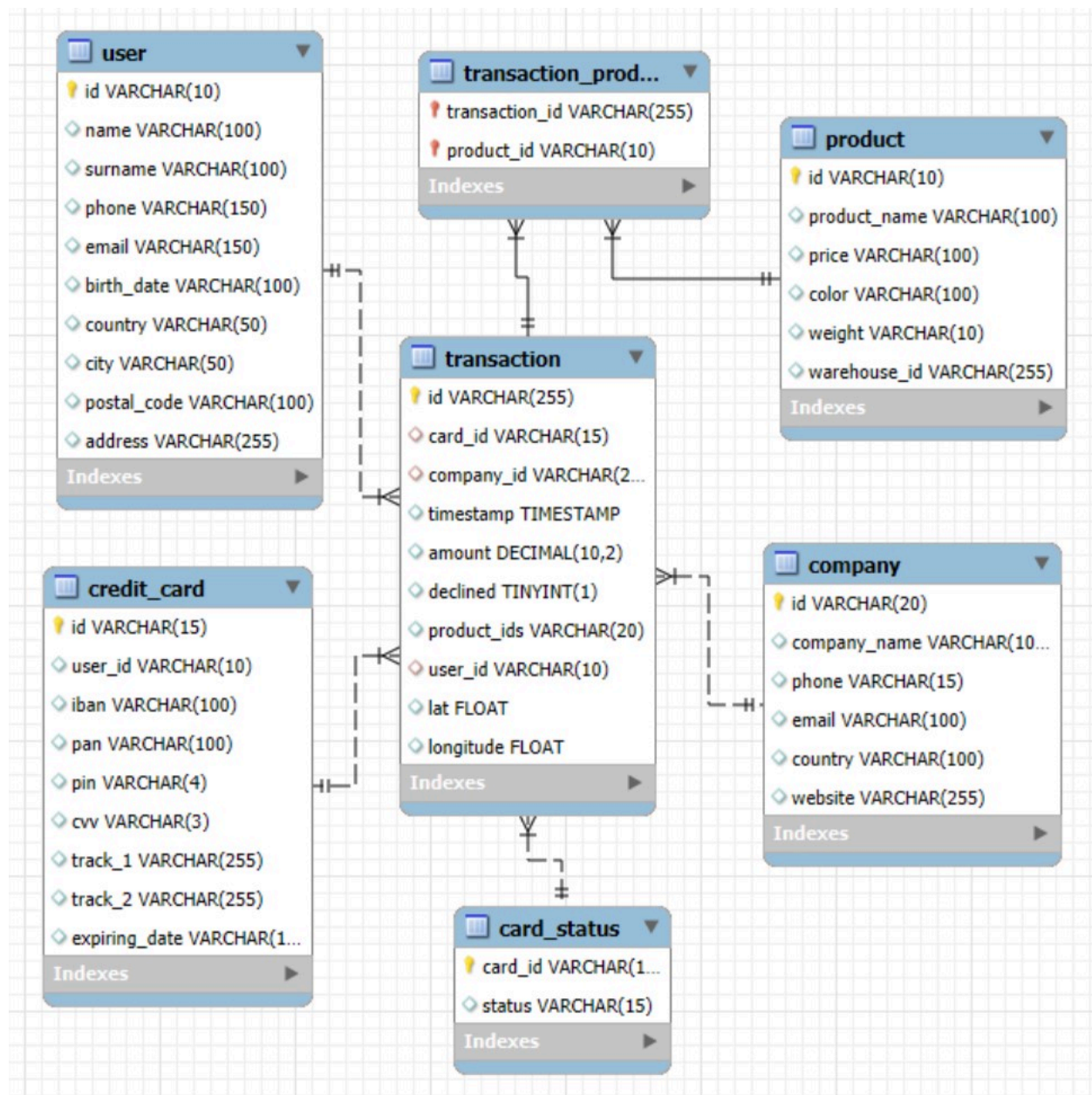
Organizando la información e insertandola en la tabla transaction_product:

```
INSERT INTO transaction_product (transaction_id, product_id)  
WITH RECURSIVE split AS (  
    Anchor part: saca el primero product ID y lo que resta de la string  
    SELECT  
        id,  
        TRIM(SUBSTRING_INDEX(product_ids, ',', 1)) AS product_id,  
        CASE WHEN INSTR(product_ids, ',') > 0  
            THEN SUBSTRING(product_ids, INSTR(product_ids, ',') + 1)  
            ELSE NULL  
        END AS rest  
    FROM transaction  
  
    UNION ALL
```

-- Recursive part: continua cortando el resto de la string, hasta que llegue a un valor NULL y para allí.

```
SELECT
    id,
    TRIM(SUBSTRING_INDEX(rest, ',', 1)) AS product_id,
    CASE WHEN INSTR(rest, ',') > 0
        THEN SUBSTRING(rest, INSTR(rest, ',') + 1)
        ELSE NULL
    END AS rest
FROM split
WHERE rest IS NOT NULL
)
SELECT id, product_id
FROM split;
```

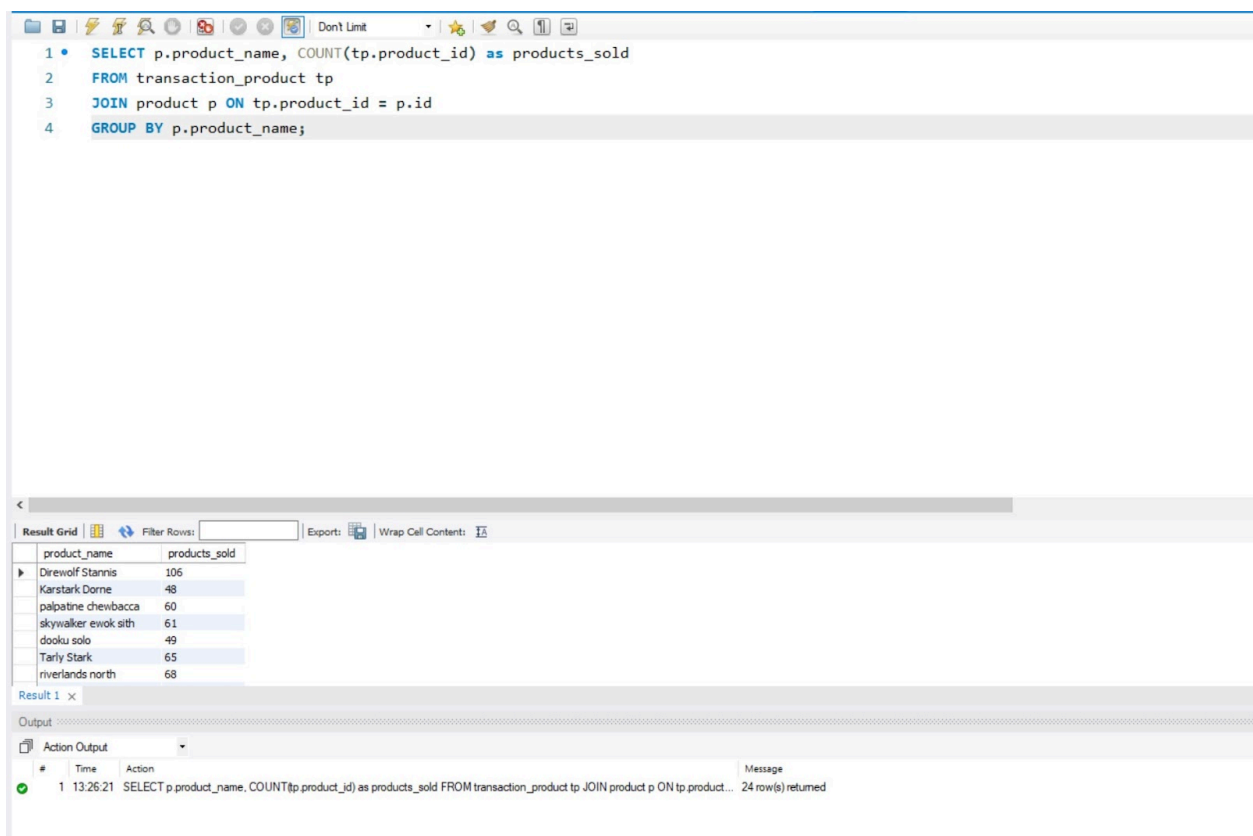
Modelo final:



EJERCICIO 1

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

We need to know the number of times each product has been sold.



The screenshot shows a SQL IDE interface. The top pane contains a SQL query:

```
1 • SELECT p.product_name, COUNT(tp.product_id) as products_sold
2 FROM transaction_product tp
3 JOIN product p ON tp.product_id = p.id
4 GROUP BY p.product_name;
```

The bottom pane displays the results in a table:

product_name	products_sold
Direwolf Stannis	105
Karstark Dome	48
palpatne chewbacca	60
skywalker ewok sith	61
dooku solo	49
Tarly Stark	65
riverlands north	68

Below the table, the 'Output' pane shows the execution message:

```
# Time Action Message
1 13:26:21 SELECT p.product_name, COUNT(tp.product_id) as products_sold FROM transaction_product tp JOIN product p ON tp.product_id = p.id 24 row(s) returned
```