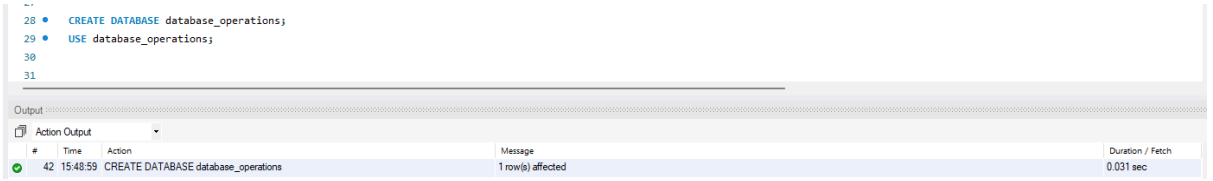
Partint d'alguns arxius CSV dissenyaràs i crearàs la teva base de dades.

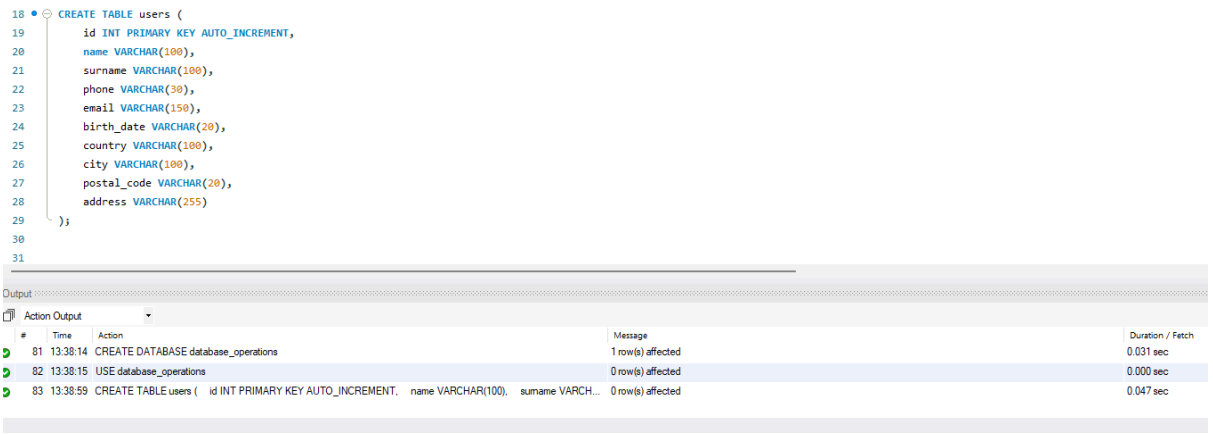
Nivell 1

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

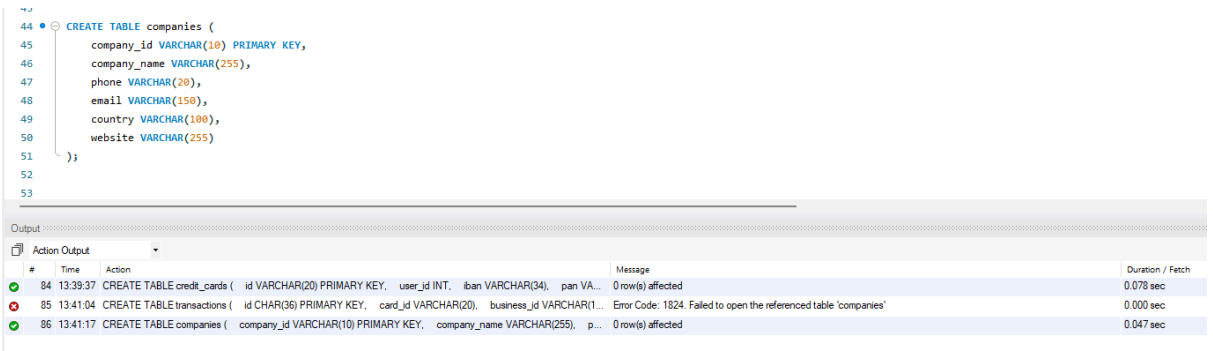
- Para comenzar creamos la DDBB llamada **database_operations** que será la que contendrá las tablas que crearemos:



- Con el comando CREATE TABLE procedemos a crear la tabla **users:**



- Creamos la tabla **companies:**



Creamos la tabla **credit card:**

```sql
CREATE TABLE credit_cards (
    id VARCHAR(20) PRIMARY KEY,
    user_id INT,
    iban VARCHAR(34),
    pan VARCHAR(30),
    pin INT,
    cvv INT,
    track1 VARCHAR(255),
    track2 VARCHAR(255),
    expiring_date VARCHAR(30)
  --   FOREIGN KEY (user_id) REFERENCES users(id)
);

```

Output

Action Output

	#	Time	Action	Message	Duration / Fetch
✔	92	17:38:00	USE database_operations	0 row(s) affected	0.000 sec
✔	93	17:38:00	CREATE TABLE users (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), sumame VARCHAR(100), phone VARCHAR(30), emai...	0 row(s) affected	0.015 sec
✔	94	17:38:00	CREATE TABLE credit_cards (id VARCHAR(20) PRIMARY KEY, user_id INT, iban VARCHAR(34), pan VARCHAR(30), pin INT, cvv INT, trac...	0 row(s) affected	0.000 sec

- La tabla **transactions** que será la que contenga las FK que la relacionará con el resto de las tablas que acabamos de crear **(users, companies y credit_cards)**.

```sql
-- CREACION TABLA TRANSACTIONS
CREATE TABLE transactions (
    id VARCHAR(50) PRIMARY KEY,
    card_id VARCHAR(20),
    business_id VARCHAR(10),
    timestamp TIMESTAMP,
    amount DECIMAL (10,2),
    declined BOOLEAN,
    product_ids VARCHAR(30),
    user_id INT,
    lat VARCHAR(50),
    longitude VARCHAR(50),
    FOREIGN KEY (card_id) REFERENCES credit_cards(id),
    FOREIGN KEY (business_id) REFERENCES companies(company_id),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Output

Action Output

	#	Time	Action	Message	Duration / Fetch
✔	174	11:31:50	CREATE TABLE credit_cards (id VARCHAR(20) PRIMARY KEY, user_id INT, iban VA...	0 row(s) affected	0.015 sec
✔	175	11:31:50	CREATE TABLE companies (company_id VARCHAR(10) PRIMARY KEY, company_name...	0 row(s) affected	0.016 sec
✔	176	11:31:50	CREATE TABLE transactions (id VARCHAR(50) PRIMARY KEY, card_id VARCHAR(20), ...	0 row(s) affected	0.046 sec

- El paso siguiente será cargar los datos de los archivos .csv a las tablas con el comando LOAD DATA INFILE:

Cargamos los datos de **users** (en este caso la tabla que creamos contendrá los datos de tres .csv (uk, usa y ca))

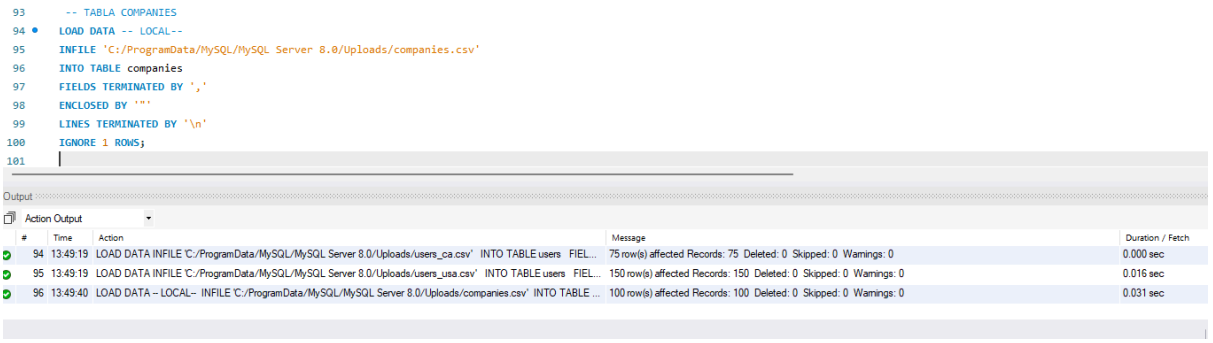
```sql
-- Cargamos datos de las tres tablas de user a una sola tabla
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS ;
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS ;
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS ;

-- Cargamos los datos de la tabla credit card
```

Output

Action Output

	#	Time	Action	Message	Duration / Fetch
✔	93	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_uk.csv' INTO TABLE users FIEL...	50 row(s) affected Records: 50 Deleted: 0 Skipped: 0 Wamings: 0	0.015 sec
✔	94	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv' INTO TABLE users FIEL...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Wamings: 0	0.000 sec
✔	95	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv' INTO TABLE users FIEL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Wamings: 0	0.016 sec

- Cargamos los datos a la tabla **companies** y hacemos lo mismo con la tabla **credit_cards**

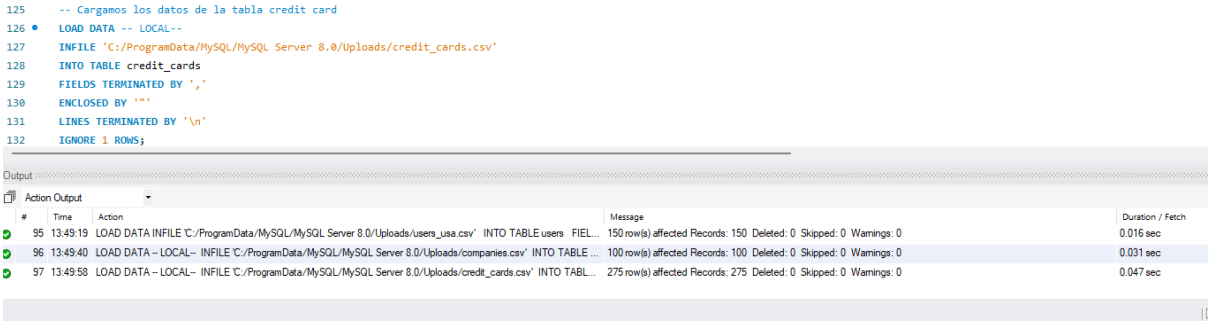


```
-- TABLA COMPANIES
LOAD DATA -- LOCAL--
INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv'
INTO TABLE companies
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
94	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_ca.csv' INTO TABLE users FIEL...	75 row(s) affected Records: 75 Deleted: 0 Skipped: 0 Warnings: 0	0.000 sec
95	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv' INTO TABLE users FIEL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
96	13:49:40	LOAD DATA -- LOCAL-- INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv' INTO TABLE ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.031 sec



```
-- Cargamos los datos de la tabla credit card
LOAD DATA -- LOCAL--
INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
95	13:49:19	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/users_usa.csv' INTO TABLE users FIEL...	150 row(s) affected Records: 150 Deleted: 0 Skipped: 0 Warnings: 0	0.016 sec
96	13:49:40	LOAD DATA -- LOCAL-- INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv' INTO TABLE ...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.031 sec
97	13:49:58	LOAD DATA -- LOCAL-- INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv' INTO TABL...	275 row(s) affected Records: 275 Deleted: 0 Skipped: 0 Warnings: 0	0.047 sec

```
-- CARGAMOS DATOS A LA TABLA TRANSACTIONS
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ';'
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

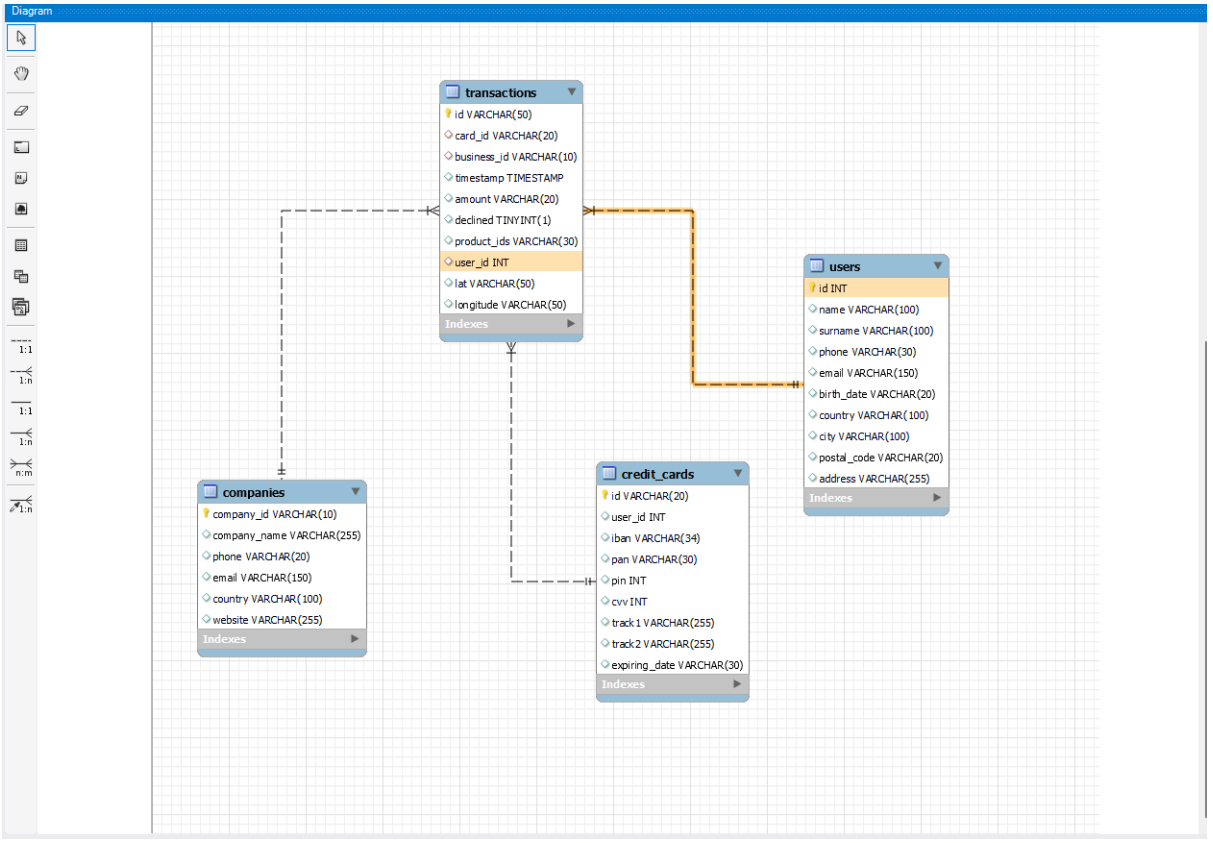
Output

Action Output

#	Time	Action	Message	Duration / Fetch
201	11:34:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv' I...	100 row(s) affected Records: 100 Deleted: 0 Skipped: 0 Warnings: 0	0.000 sec
202	11:34:01	LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv...	587 row(s) affected Records: 587 Deleted: 0 Skipped: 0 Warnings: 0	0.063 sec

- Con estos **datos** ya cargados podemos ver el diagrama que ha quedado de la siguiente manera siendo la tabla de hechos la tabla **transactions** y las tablas de dimensiones la

tabla users, companies y credit_cards.



- Exercici 1

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

- Para saber cuales fueron los usuarios que ha realizado más de 30 transacciones seleccionamos los usuarios de la tabla **users** y obtenemos los datos de la tabla **transactions** con una subconsulta esto nos permite contar la cantidad de transacciones por id de usuario
- Para realizar esta consulta podemos hacerla con JOIN que traera los datos que coinciden de las tablas transactions y users y con la funcion HAVING COUNT contamos los datos agrupados por id y esto nos traera los resultados

```sql
SELECT u.name,u.id,
COUNT(t.id) AS CantTransactions
FROM users u
JOIN transactions t ON t.user_id=u.id
GROUP BY u.name,u.id
HAVING COUNT(t.id) > 30;
```

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

name	id	CantTransactions
Lynn	92	39
Ocean	267	52
Hedwig	272	76
Kenyon	275	48

Result 49 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
356	11:14:04	SELECT * FROM database_operations.card_status LIMIT 0, 2000	275 row(s) returned	0.000 sec / 0.000 sec
357	11:31:38	SELECT u.id, u.name FROM users u WHERE (SELECT COUNT(*) FROM transactions t WH...	4 row(s) returned	0.016 sec / 0.000 sec
358	11:31:42	SELECT u.name,u.id, COUNT(t.id) AS CantTransactions FROM users u JOIN transactions t ...	4 row(s) returned	0.016 sec / 0.000 sec

- Exercici 2

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

Para saber la media de amount por IBAN de las tarjetas de credito a la compañia Donec Ltd necesitamos los datos de la tabla **credit cards** (IBAN), **transactions** (para obtener los datos de amount)y **companies** de donde obtendremos el nombre de la compañia que coincide con la que buscamos, obtenemos estos datos mediante un JOIN de las tres tabas y luego filtramos el nombre con WHERE y agrupamos por IBAN.

```sql
SELECT cc.iban, round(avg(t.amount),2) AS Media_Amount
FROM credit_cards cc
JOIN transactions t ON cc.id=t.card_id
JOIN companies c ON c.company_id=t.business_id
WHERE c.company_name ="Donec Ltd"
GROUP BY cc.iban;
```

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

iban	Media_Amount
PT87806228135092429456346	203.72

Result 8 | Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
229	12:23:40	SELECT u.name,u.id, COUNT(t.id) AS CantTransactions FROM users u JOIN transactions t ON t.user_id=u.id GROUP BY u.name,u.id HAVING COUNT (t.id) > 3...	Error Code: 1630. FUNCTION database_operations.COUNT does not exist. ...	0.000 sec
230	12:25:37	SELECT u.id, u.name FROM users u WHERE (SELECT COUNT(*) FROM transactions t WHERE t.user_id = u.id) > 30 LIMIT 0, 1000	4 row(s) returned	0.000 sec / 0.000 sec
231	12:28:23	SELECT cc.iban, round(avg(t.amount),2) AS Media_Amount FROM credit_cards cc JOIN transactions t ON cc.id=t.card_id JOIN companies c ON c.company_id=t...	1 row(s) returned	0.000 sec / 0.000 sec

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:

- Para esto creamos la tabla **card_status** donde card_id es la PK y es la columna que referencia a la otra tabla (**credit_cards**) como FK para asegurar que cada id exista en la tabla **credit_cards**, y la otra columna tenemos active_status que nos indicará si la tarjeta cumple o no con la condición pedida (en este caso si las últimas tres transacciones fueron declined la tarjeta estará inactiva de lo contrario su condición será 'activa'.

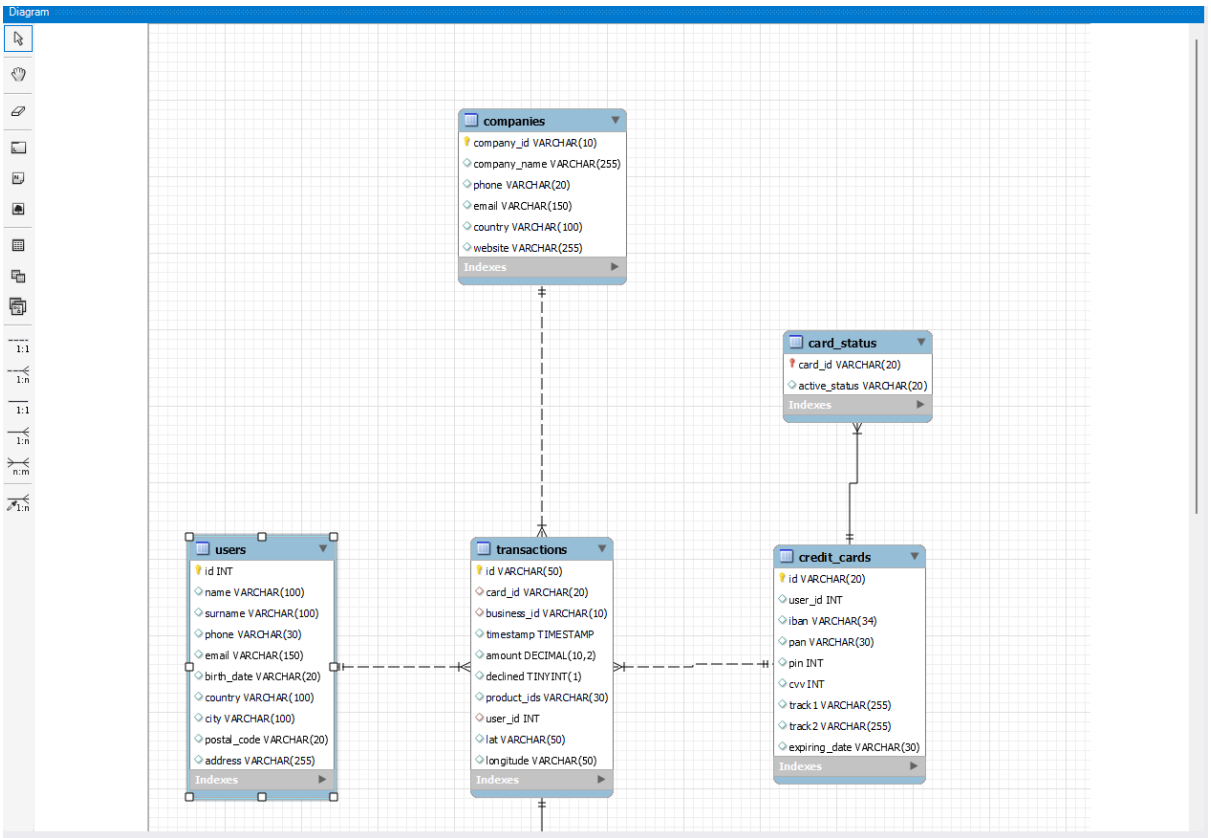
```
36 • CREATE TABLE card_status (
37       card_id VARCHAR(20) PRIMARY KEY,
38       active_status VARCHAR(20),
39       FOREIGN KEY (card_id) REFERENCES credit_cards(id)
40   );
41
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
394	10:12:39	SELECT SUM(t.declined) FROM transactions t WHERE t.card_id = cc.id ORDER BY t.times...	Error Code: 1054. Unknown column 'cc.id' in 'where clause'	0.000 sec
395	10:27:35	DROP TABLE 'database_operations'.'card_status'	0 row(s) affected	0.032 sec
396	10:30:02	CREATE TABLE card_status (card_id VARCHAR(20) PRIMARY KEY, active_status VARC...	0 row(s) affected	0.063 sec

- Con esta tabla creada nuestro diagrama EER se verá así, de esta manera con la creación de la nueva tabla se transforma en un diagrama modelo copo de nieve ya que la tabla card_status no se relaciona con transactions sino con la tabla de dimensiones card_status.



- Para el siguiente paso que es la selección de datos a insertar en la tabla realizamos los siguientes pasos:

Se recorre la tabla credit_cards (con alias cc) para evaluar cada tarjeta y utilizando ROW NUMBER le asignamos un número a cada transacción y de esta manera nos quedamos sólo con tres que son las que necesitamos; las ordenamos de forma descendente según la columna timestamp,de la más reciente a la más antigua. Mediante la operación SUM se calcula la suma de la columna declined. Si una transacción fue rechazada, el valor de declined es 1 y, de lo contrario, es 0. Usamos COALESCE(..., 0) para devolver 0 en caso de que la subconsulta no encuentre transacciones (evitando así valores nulos). Utilizamos el LEFT JOIN para que aparezcan todas las tarjetas incluso si no tienen transacciones y conectamos cada tarjeta con su información en transacciones.

Mediante el uso de CASE determinamos el estado siendo que la suma de los valores de declined de las últimas tres transacciones es igual a 3, significa que las tres transacciones fueron rechazadas, por lo que la tarjeta se marca como "No Active" en cualquier otro caso, la tarjeta se considera "Active". Finalmente el resultado se inserta en la tabla **card_status.** Mediante la inserción de datos de esta manera evaluamos para cada tarjeta el estado de las últimas tres transacciones.

```sql
INSERT INTO card_status (card_id, active_status)
SELECT
    cc.id,
    CASE
        WHEN COALESCE(a.sum_declined, 0) = 3 THEN 'No Activo'
        ELSE 'Activo'
    END AS active_status
FROM credit_cards cc
LEFT JOIN (
    SELECT
        card_id,
        SUM(declined) AS sum_declined
    FROM (
        SELECT
            card_id,
            declined,
            ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS rn
        FROM transactions
    ) AS last_three
    WHERE rn <= 3
    GROUP BY card_id
) AS a ON a.card_id = cc.id;


SELECT version();
```

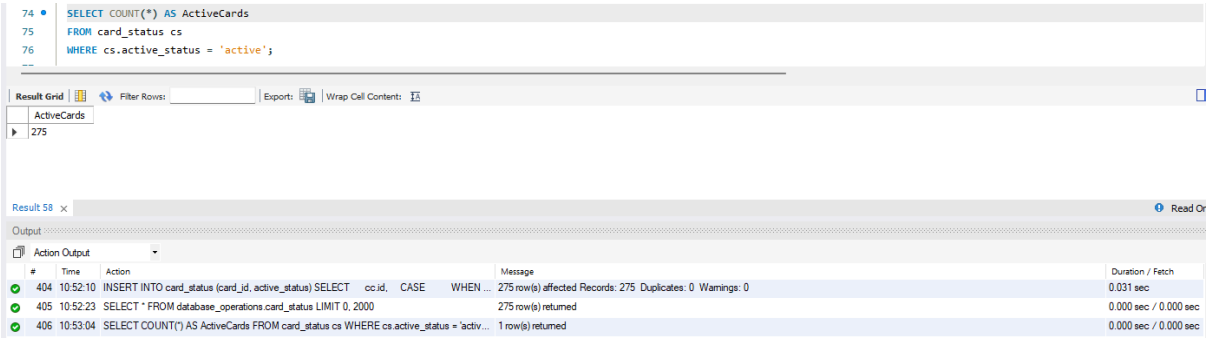
```
Output
Action Output
#  Time      Action                                                                                  Message                                                                  Duration / Fetch
465 12:05:17 CREATE TABLE card_status ( card_id VARCHAR(20) PRIMARY KEY,  active_status VARC...      0 row(s) affected                                                        0.047 sec
466 12:05:27 INSERT INTO card_status (card_id, active_status) SELECT    cc.id,    CASE        WHEN ...  276 row(s) affected Records: 276  Duplicates: 0  Warnings: 0  0.016 sec
```

Exercici 1

Quantes targetes estan actives?

- Aquí realizamos un COUNT (*) de la cantidad de tarjetas cuyos campos son = a la condición introducida anteriormente ‘active’ lo que nos daría como resultado el total de tarjetas activas, en este caso podemos ver que todas las tarjetas se encuentran activas

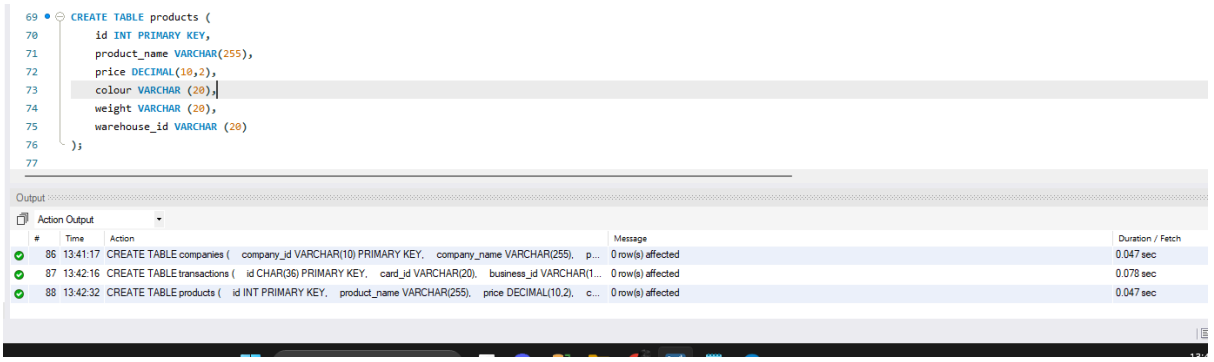
ya que no tenemos ninguna que coincida con el filtro de ùltimas tres transacciones declinadas.



Nivell 3

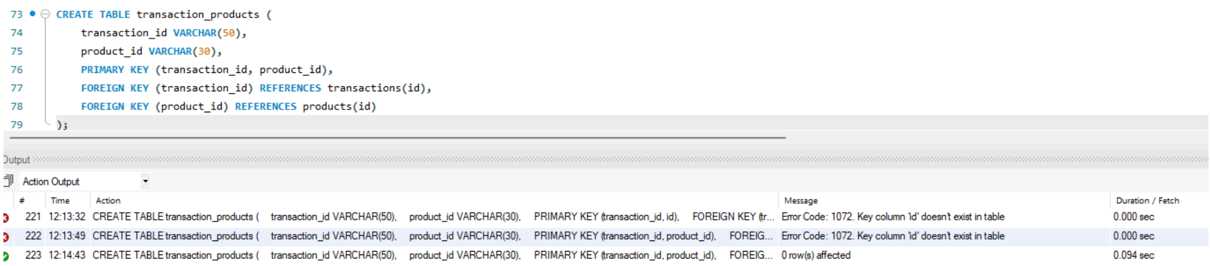
Crea una taula amb la qual puguem unir les dades del nou arxiu products.csv amb la base de dades creada, tenint en compte que des de transaction tens product_ids. Genera la següent consulta:

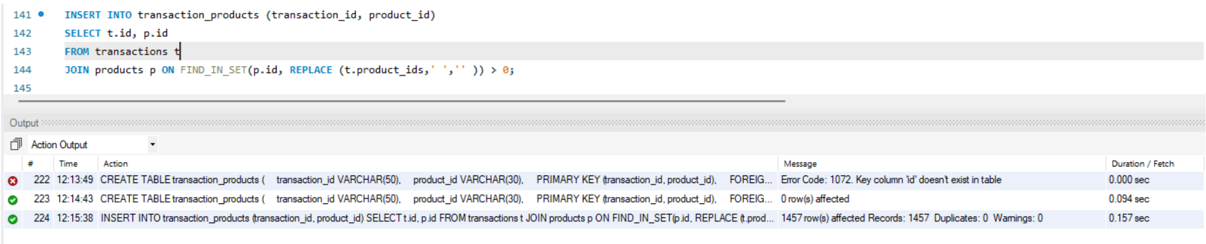
- Para crear una tabla que una la tabla **products** con **transactions** debemos realizar los siguientes pasos:
- Crear y cargar los datos de la tabla **products,** cuando cargamos los datos a la tabla debemos cambiar el formato de la columna price ya que contiene el símbolo $ para esto realizamos un CAST y REPLACE para poder corregir esto.



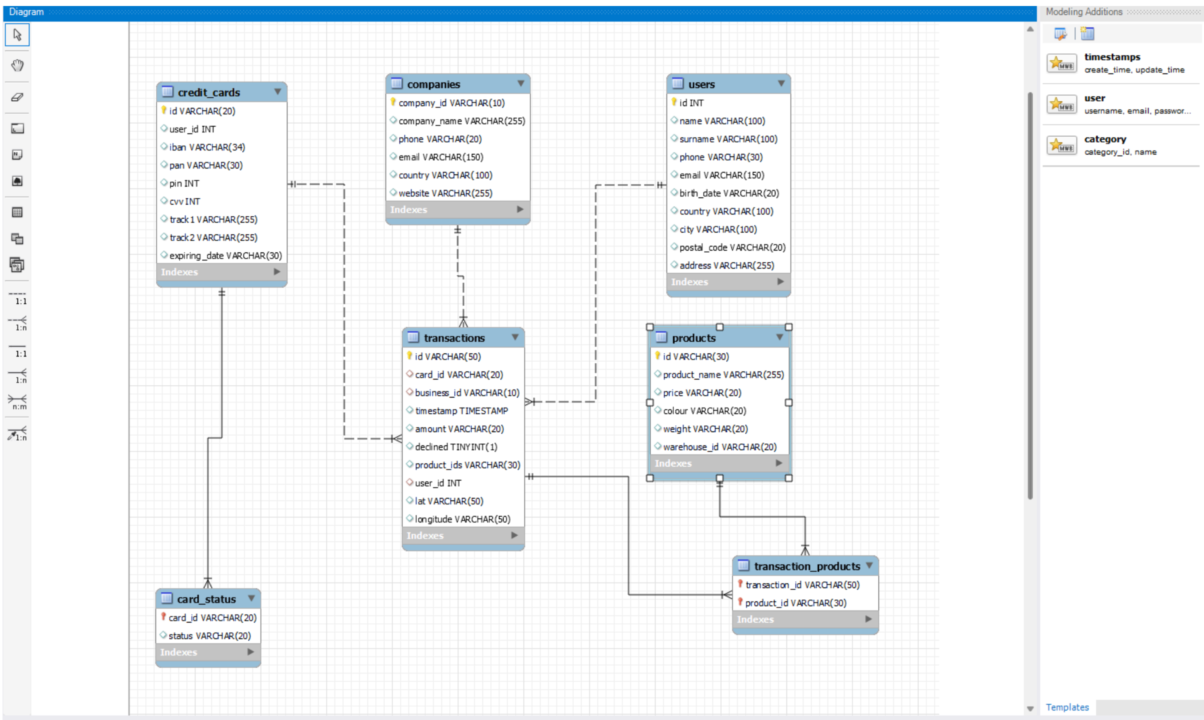
- La siguiente parte es crear la tabla intermedia **transaction_products** para gestionar y evitar la relacion n:n entre **transactions y products** , esta tabla tendrá una PK

compuesta entre transaction_id y product_id ya que cada fila contendrá un producto y a la vez contiene dos columnas (FKs) en este caso transaction_id y product_id; como la tabla transactions posee los **products_ids** separados por coma debemos realizar un FIND_IN_SET para extraer los IDs de productos dentro de la transacción y busca si el ID del producto (`p.id`) está dentro de esa lista, mediante el REPLACE lo que hacemos es eliminar los espacios entre los productos y si el producto es mayor a 0 significa que esta dentro de esa transacción y se insertará en la fila en la tabla que acabamos de crear.





Con estos datos nuestro diagrama se ve de la siguiente manera:



Exercici 1

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

- Para conocer el nombre de veces que se ha vendido cada producto necesitamos los datos de la tabla que acabamos de crear y de la tabla **products** para obtener el nombre de los productos, para esta consulta contamos la cantidad de veces que aparece el **product_id** en la tabla **transaction_products** y agrupamos los resultados por nombre.