IT ACADEMY **Sprint 4 . Yoneidy Galindez**

Descripción de las tablas

transaction

id: es el identificador de cada transacción y clave primaria de la tabla.

card_id: id de la tarjeta usada en la transacción.

business_id: identificador del negocio donde se realizó la transacción.

timestamp: fecha y hora cuando se realizó la transacción.

amount: monto de las transacciones.

declined: indicador de si la transacción fue rechazada(1) ó aceptada(0)

product_ids: productos comprados en la transacción.

user_id: identificador del usuario que realizó la transacción.

latitud: geográfica donde se realizó la transacción.

Longitude: longitud geográfica donde ocurrió la transacción.

users

id: identificador del usuario.
nombre: nombre del usuario.
surname :apellido del usuario.
phone: número de teléfono del usuario(puede incluir codigo de pais)
email: correo electrónico del usuario.
birth_date: fecha de nacimiento del usuario.
country: país de residencia.
ciudad: ciudad de residencia.
postal_code: código postal del domicilio del usuario.
address: dirección completa del usuario.
región: etiqueta al usuario según procedencia como americano o europeo.

companies

company_id: identificador de la empresa.
company_name: nombre de la empresa.
phone: numero de telefono de la empresa.
email: correo electrónico contacto de la empresa.
country: país de la empresa.
website: sitio web de la empresa.

credit_cards

id:identificador de la tarjeta de crédito.
user_id: identificador del usuario de la tarjeta(relacion con tabla users)
iban: código iban asociado a la tarjeta.

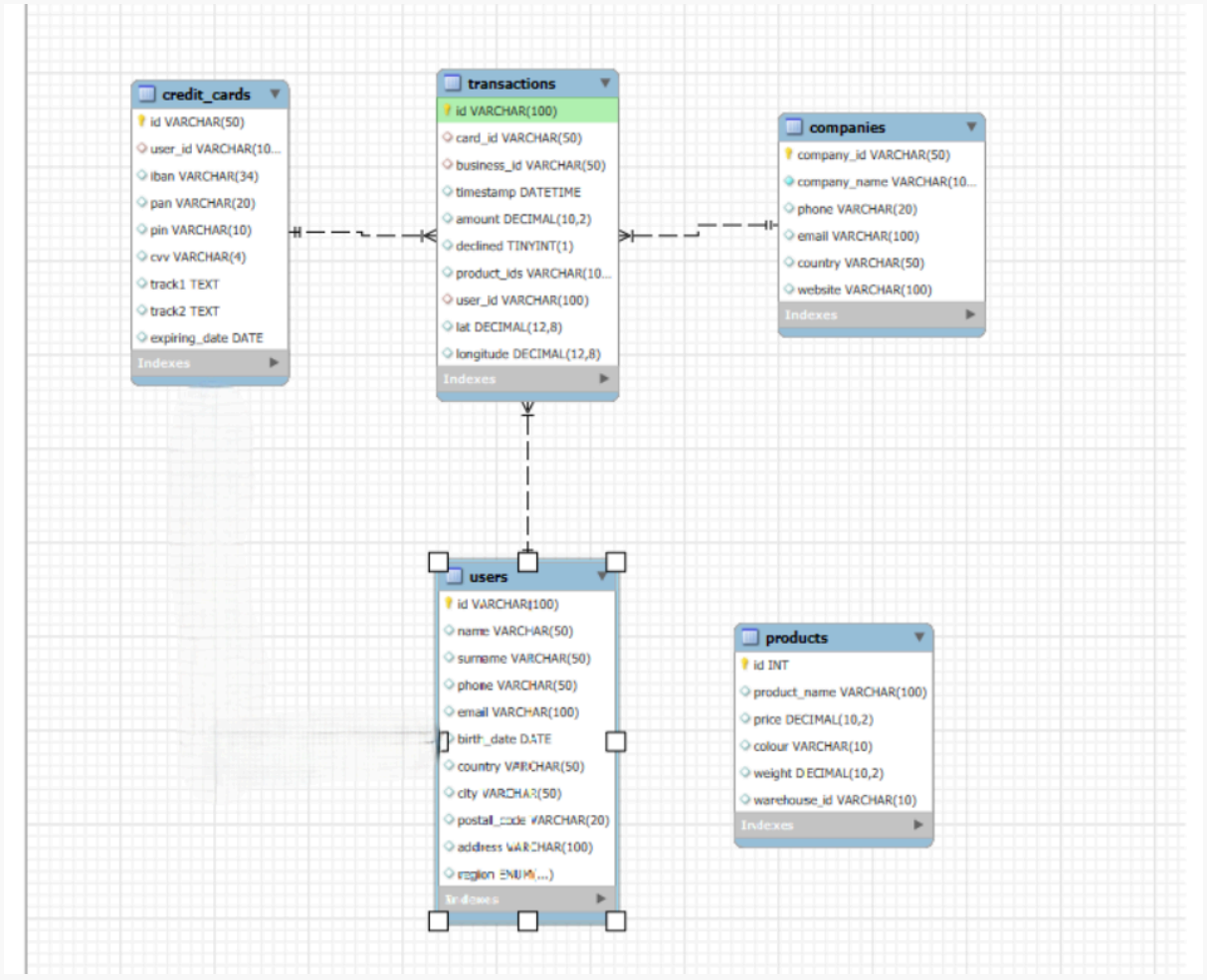
pan:número de la tarjeta de crédito.
pin: número de identificación personal de la tarjeta.
cvv: código de verificación de la tarjeta.
track1: información de la banda magnética 1.
track2: información de la banda magnética 2.
expiring_date: fecha de expiración de la tarjeta.

products
id: identificador del producto.
product_name: nombre del producto.
precio:precio del producto.
colour: color del producto.
weight: peso del producto.
warehouse_id: identificador del almacén donde se guarda el producto.

transaction_products
transaccion _id: identificador de la transacción (foreign key de transaction.id) clave primaria compuesta con product id.
product_id: identificador del producto(foreign key a products.id) clave primaria compuesta con products.id.

status_credit_card
card_id: identificador de la tarjeta de crédito. clave primaria y foránea de credit_cards(id)
status_card : estado de la tarjeta.(activa o inactiva)

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:




```
1 •  CREATE DATABASE IF NOT EXISTS Ecommerce;
2 •  USE Ecommerce;
3
4
```

Creamos la base de datos llamada Ecommerce.

```sql
-- Tabla: companies
CREATE TABLE companies (
    company_id VARCHAR(50) PRIMARY KEY,
    company_name VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    email VARCHAR(100),
    country VARCHAR(50),
    website VARCHAR(100)
);
```

Creamos la tabla companies.

copiamos la ruta de acceso, copiamos en mysql y luego realizamos la descarga de la base de datos usando load infile

```sql
SHOW variables like 'secure_file_priv';

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/companies.csv"
INTO TABLE companies
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(company_id, company_name, phone, email, country, website);
```

Se especifica que el nombre de la tabla donde se insertan los datos es companies, los campos del archivo están separados por coma(fields terminated by), los valores de los campos están cerrados por comillas dobles(enclosed by), está terminado en un salto de línea(Lines terminated by), ignora la primera fila(ignore 1 rows) porque casi siempre están los encabezados y luego se especifica orden de las columnas del archivo. Haremos el mismo procedimiento con las demás tablas de la base de datos.

```sql
CREATE TABLE products (
    id INT PRIMARY KEY,
    product_name VARCHAR(100),
    price VARCHAR(20),
    colour VARCHAR(10),# formato hexadecimal, como #7c7c7c
    weight DECIMAL(10,2),
    warehouse_id VARCHAR(10)
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv"
INTO TABLE products
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, product_name, price, colour, weight, warehouse_id);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	$161.11	#7c7c7c	1.00	WH-4
2	Tarly Stark	$9.24	#919191	2.00	WH-3

Se puede observar que la columna Price de tipo Varchar puede leer símbolos pero debemos transformarla para poder realizar operaciones numéricas con esta columna.

```sql
UPDATE products
SET price = REPLACE(price, '$', '');
ALTER TABLE products
MODIFY COLUMN price DECIMAL(10,2);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

id	product_name	price	colour	weight	warehouse_id
1	Direwolf Stannis	161.11	#7c7c7c	1.00	WH-4
2	Tarly Stark	9.24	#919191	2.00	WH-3

se realiza un update para reemplazar el símbolo por un espacio y luego se modifica la columna a tipo Decimal.

```sql
-- Tabla: american_users
CREATE TABLE american_users (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(50),
    surname VARCHAR(50),
    phone VARCHAR(50),
    email VARCHAR(100),
    birth_date VARCHAR(50),
    country VARCHAR(50),
    city VARCHAR(50),
    postal_code VARCHAR(20),
    address VARCHAR(100)
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/american_users.csv"
INTO TABLE american_users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id,name, surname, phone, email, birth_date,country,city,postal_code,address);

```

```sql
CREATE TABLE european_users (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(50),
    surname VARCHAR(50),
    phone VARCHAR(50),
    email VARCHAR(100),
    birth_date VARCHAR(50),
    country VARCHAR(50),
    city VARCHAR(50),
    postal_code VARCHAR(20),
    address VARCHAR(100)
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/european_users.csv"
INTO TABLE european_users
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id,name, surname, phone, email, birth_date,country,city,postal_code,address);
```

```sql
CREATE TABLE users (
    id VARCHAR(100) PRIMARY KEY,
    name VARCHAR(50),
    surname VARCHAR(50),
    phone VARCHAR(50),
    email VARCHAR(100),
    birth_date VARCHAR(50),
    country VARCHAR(50),
    city VARCHAR(50),
    postal_code VARCHAR(20),
    address VARCHAR(100),
    region ENUM('american', 'european')  -- para saber de qué tabla vino
);
```

Para unir las dos tablas de usuarios hacemos lo siguiente
creamos una columna extra llamada región que usa Enum

ENUM, o enumeración, es un tipo de dato en MySQL diseñado para almacenar un valor de un conjunto predefinido de constantes de cadena. Las columnas ENUM son extremadamente útiles cuando conoces los posibles valores que puede tomar un campo, como los días de la semana, los estados de un pedido, categorías fijas, entre otros.

```sql
-- Insertar usuarios americanos
INSERT INTO users
SELECT id, name, surname, phone, email, birth_date, country, city, postal_code, address, 'american'
FROM american_users;

-- Insertar usuarios europeos
INSERT INTO users
SELECT id, name, surname, phone, email, birth_date, country, city, postal_code, address, 'european'
FROM european_users;
```

con el comando insert se crea la nueva tabla de usuarios.

```sql
SELECT * FROM users LIMIT 10;
SELECT COUNT(*) FROM users;


#eliminar las tablas european y american users.
DROP TABLE american_users;
DROP TABLE european_users;
```

Luego se eliminan las dos tablas american y european users, ya que las tenemos almacenadas a nuestra nueva tabla. la ventaja de realizar esto es que por ello pudimos

crear la nueva tabla que etiqueta de donde provienen los usuarios y así poder realizar futuras consultas.

```sql
#cambiar el tipo de fecha
UPDATE users
SET birth_date = STR_TO_DATE(birth_date, '%b %d, %Y');
-- Cambiar tipo de columna
ALTER TABLE users MODIFY COLUMN birth_date DATE;
SELECT birth_date FROM users;
```

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

birth_date
1985-11-17
1984-04-30
1989-09-15
1970-05-17
1994-03-04

También cambiamos la fecha ya que estaba en el formato(Nov 17, 1985) y este formato no es compatible con Date , por ello lo transformamos.

```sql
-- Tabla: credit_cards
CREATE TABLE credit_cards (
    id VARCHAR(50) PRIMARY KEY,
    user_id VARCHAR(100),
    iban VARCHAR(34),
    pan VARCHAR(20),
    pin VARCHAR(10),
    cvv VARCHAR(4),
    track1 TEXT,
    track2 TEXT,
    expiring_date VARCHAR(50)
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/credit_cards.csv"
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, user_id,iban,pan,pin,cvv, track1,track2,expiring_date);

SELECT * FROM credit_cards;
```

Hemos cargado la tabla credit cards y luego realizamos una actualización en la variable expiring date.

```sql
UPDATE credit_cards
SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%Y');
ALTER TABLE credit_cards MODIFY COLUMN expiring_date DATE;
SELECT * FROM credit_cards;

ALTER TABLE credit_cards
ADD FOREIGN KEY (user_id) REFERENCES users(id);
```

se cambió la fecha a Date.y también se agregó la foreign key.

```sql
CREATE TABLE transactions (
    id VARCHAR(100) PRIMARY KEY,
    card_id VARCHAR(50),
    business_id VARCHAR(50),
    timestamp DATETIME,
    amount DECIMAL(10,2),
    declined BOOLEAN,
    product_ids VARCHAR(100),
    user_id INT,
    lat DECIMAL(12,8),
    longitude DECIMAL(12,8)
);

LOAD DATA INFILE "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transactions.csv"
INTO TABLE transactions
FIELDS TERMINATED BY ';'
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(id, card_id, business_id, timestamp, amount, declined, product_ids, user_id, lat, longitude);
```

luego a la tabla transaction se agrega las claves foráneas

```sql
SELECT * FROM transactions;
ALTER TABLE transactions MODIFY COLUMN user_id VARCHAR(100);

-- transactions.user_id → users.id
ALTER TABLE transactions
ADD FOREIGN KEY (user_id) REFERENCES users(id);

-- transactions.card_id → credit_cards.id
ALTER TABLE transactions
ADD FOREIGN KEY (card_id) REFERENCES credit_cards(id);

-- transactions.business_id → companies.company_id
ALTER TABLE transactions
ADD FOREIGN KEY (business_id) REFERENCES companies(company_id);
```

Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 80 transacciones utilizando al menos 2 tablas.

```
225 •  SELECT u.id,u.name,u.surname
226    FROM users u
227    WHERE u.id IN  (SELECT t.user_id
228                FROM transactions t
229                WHERE declined=0
230                GROUP BY t.user_id
231                HAVING COUNT(t.id) > 80);
232
233
234
235
236
237
238
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	surname
185	Molly	Gilliam
289	Dxwgi	Hwcru
318	Bnyr	Astuw
NULL	NULL	NULL

users 23

Output

Action Output

#	Time	Action	Message
1	14:48:29	SELECT u.id,u.name,u.surname FROM users u WHERE u.id IN (SELECT t.user_id FROM transactions ...	3 row(s) returned

Ejercicio 2

Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.

```
235
236 •   SELECT cc.iban,AVG(t.amount) AS Avg_amount,C.company_name
237     FROM credit_cards cc
238     JOIN transactions t ON cc.id=t.card_id
239     JOIN companies C ON C.company_id=t.business_id
240     WHERE company_name='Donec Ltd' AND declined=0
241     GROUP BY cc.iban,C.company_name
242     ORDER BY Avg_amount desc;
243
244
245
246
247
```

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

iban	Avg_amount	company_name
XX3830178139196201993663 52	680.690000	Donec Ltd
XX6377063573975703949739 13	680.010000	Donec Ltd
XX9713939714652922023122 59	645.460000	Donec Ltd
XX1718471169288923759693 07	628.890000	Donec Ltd
XX2254246388185424062235 75	608.680000	Donec Ltd

Result 25

Output

Action Output

#	Time	Action	Message
1	14:57:20	SELECT cc.iban,AVG(t.amount) AS Avg_amount,C.company_name FROM credit_cards cc JOIN transa...	370 row(s) returned

Nivel 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las últimas tres transacciones fueron declinadas y genera la siguiente consulta:

Ejercicio 1

¿Cuántas tarjetas están activas?

```
#CREAMOS UNA NUEVA TABLA STATUS_CREDIT_CARD
CREATE TABLE status_credit_card (
    card_id VARCHAR(50) PRIMARY KEY,
    status_card VARCHAR(50) NOT NULL,
    FOREIGN KEY (card_id) REFERENCES credit_cards(id)
    );

#CARGAMOS LA NUEVA TABLA CON LAS CONDICIONES CORRESPONDIENTES
INSERT INTO status_credit_card (card_id, status_card)
SELECT card_id,
       CASE
           WHEN SUM(CASE WHEN declined THEN 1 ELSE 0 END) = 3 THEN 'Inactiva'
           ELSE 'Activa'
       END AS card_status
FROM (
    SELECT card_id, declined,
           ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS latest_transactions
    FROM transactions
) AS t
WHERE latest_transactions <= 3
GROUP BY card_id;
```

Output

Action Output

#	Time	Action	Message
1	14:57:20	SELECT cc.iban,AVG(t.amount) AS Avg_amount,C.company_name FROM credit_cards cc JOIN transa...	370 row(s) returned
2	15:05:58	CREATE TABLE status_credit_card (card_id VARCHAR(50) PRIMARY KEY, status_card VARCH...	0 row(s) affected
3	15:16:09	INSERT INTO status_credit_card (card_id, status_card) SELECT card_id, CASE WHEN SU...	5000 row(s) affected Records: 5000 Duplicates: 0 Warnings: 0

Nivel 3

Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde transaction tienes product_ids.

Ejercicio 1

Necesitamos conocer el número de veces que se ha vendido cada producto.

```sql
CREATE TABLE transaction_products (
    transaction_id VARCHAR(100),
    product_id INT,
    PRIMARY KEY (transaction_id, product_id),
    FOREIGN KEY (transaction_id) REFERENCES transactions(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);

INSERT INTO transaction_products (transaction_id, product_id)
SELECT transactions.id AS transactions_id, products.id AS products_id
FROM transactions
JOIN products
ON FIND_IN_SET(products.id, REPLACE(transactions.product_ids, ' ', '')) > 0;
```

La función en MySQL se utiliza para buscar una cadena dentro de una lista de cadenas separadas por comas.FIND_IN_SET()

```sql
SELECT product_id, COUNT(*) AS num_ventas
FROM transaction_products
JOIN transactions ON transaction_id = transactions.id
WHERE declined = 0
GROUP BY product_id
ORDER BY num_ventas DESC;
```

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

product_id	num_ventas
52	2642
29	2627
21	2603
16	2602
33	2593
87	2591

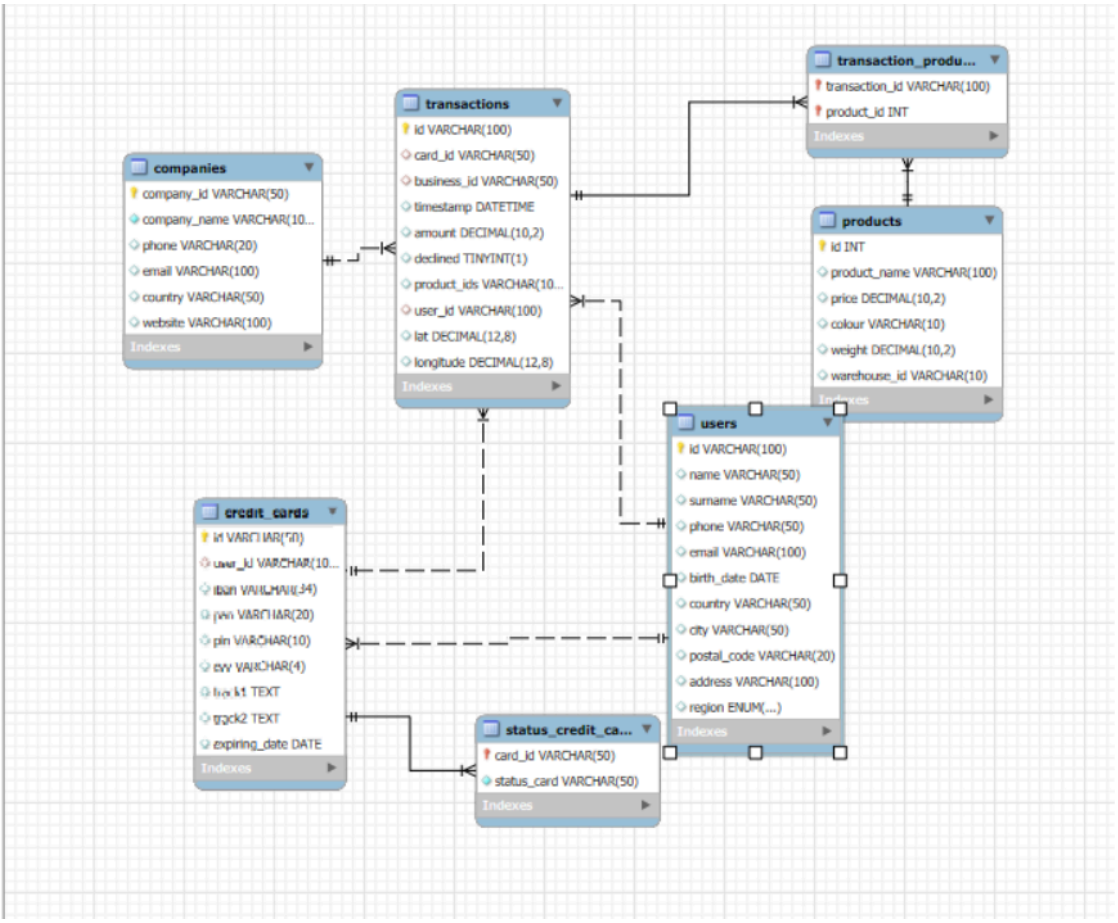
Result 47

Output

Action Output

#	Time	Action	Message
1	15:50:54	SELECT product_id, COUNT(*) AS num_ventas FROM transaction_products JOIN transactions ON tran...	100 row(s) returned

por último se realiza la consulta.



Relación entre las tablas

users -- credit cards 1:N
Un usuario puede tener muchas tarjetas de crédito.

Credit_cards--status_credit_cards 1.1
Cada tarjeta tiene un estatus asociado.

Credit_cards- transaction 1:N
una tarjeta puede realizar muchas transacciones.

Companies- transactions 1:N
cada transacción se realiza con una empresa.

transaction- transaction_products 1:N
Una transacción puede incluir varios productos.

products-transaction_products 1:N
Un producto puede estar presente en varias transacciones.