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:

A continuación explico como hemos generado todas estas tablas:

1) Procedemos a crear la tabla company

```
CREATE TABLE company (
  company_id VARCHAR(15) PRIMARY KEY,
  company_name VARCHAR(255),
  phone VARCHAR(15),
  email VARCHAR(100),
  country VARCHAR(100),
  website VARCHAR(255)
);
```

2) Creamos la tabla credit_card

```
CREATE TABLE credit_card (
  id VARCHAR(20) PRIMARY KEY,
  user_id INT,
  iban VARCHAR(50),
  pan VARCHAR(30),
  pin VARCHAR(4),
  cvv INT,
  track1 VARCHAR(255),
  track2 VARCHAR(255),
  expiring_date VARCHAR(10)
);
```

3) Creamos la tabla product

```sql
#Creamos la tabla product
CREATE TABLE product (
  id INT PRIMARY KEY,
  product_name VARCHAR(255),
  price VARCHAR(15),
  colour VARCHAR(10),
  weight DECIMAL(5, 2),
  warehouse_id VARCHAR(10)
);
```

- Una vez ya hemos añadido toda la información, procedemos a borrar el símbolo del $ y cambiarle el formato al campo, ahora pasará a ser decimal. Ahora tendremos la columna de price totalmente limpia para poder operar con ella.

```sql
UPDATE product
SET price = CAST(REPLACE(price, '$', '') AS DECIMAL(10, 2))
WHERE price LIKE '$%';
```

4) Creamos la tabla transaction

```sql
CREATE TABLE transaction (
  id VARCHAR(255) PRIMARY KEY,
  card_id VARCHAR(15),
  business_id VARCHAR(20),
  timestamp TIMESTAMP,
  amount DECIMAL(10, 2),
  declined TINYINT(1),
  product_ids VARCHAR(255),
  user_id INT,
  lat FLOAT,
  longitude FLOAT
);
```

5) Creamos la tabla data_user

```sql
CREATE TABLE data_user (
  id INT PRIMARY KEY,
  name VARCHAR(100),
  surname VARCHAR(100),
  phone VARCHAR(150),
  email VARCHAR(150),
  birth_date VARCHAR(100),
  country VARCHAR(150),
  city VARCHAR(150),
  postal_code VARCHAR(100),
  address VARCHAR(255)
);
```

6) Necesitamos generar una tabla nueva, la llamaremos "transactions_with_products".

Esta tabla nueva la utilizaremos cuando necesitemos hacer busquedas por los productos comprados individualmente, ya que en products_id, salen todos los productos comprados por cada transaccion. En este caso solo necesitaremos extraer la info de cada producto a que id transaccion hara referencia. Por ejemplo si dejaramos el amount, cada producto arrastraria el amount de la compra total y tendriamos un numero totalmente erroneo si calcularamos el amount a traves de esta nueva tabla.

```sql
CREATE TABLE transactions_with_products (
    id VARCHAR(255),
    product_id VARCHAR(255)
);
```

7) En este apartado me he tenido que apoyar con GPT para poder extraer la informacion de la tabla transactions de tal manera que me extraiga cada products de products_id y no pierda el resto de la informacion.

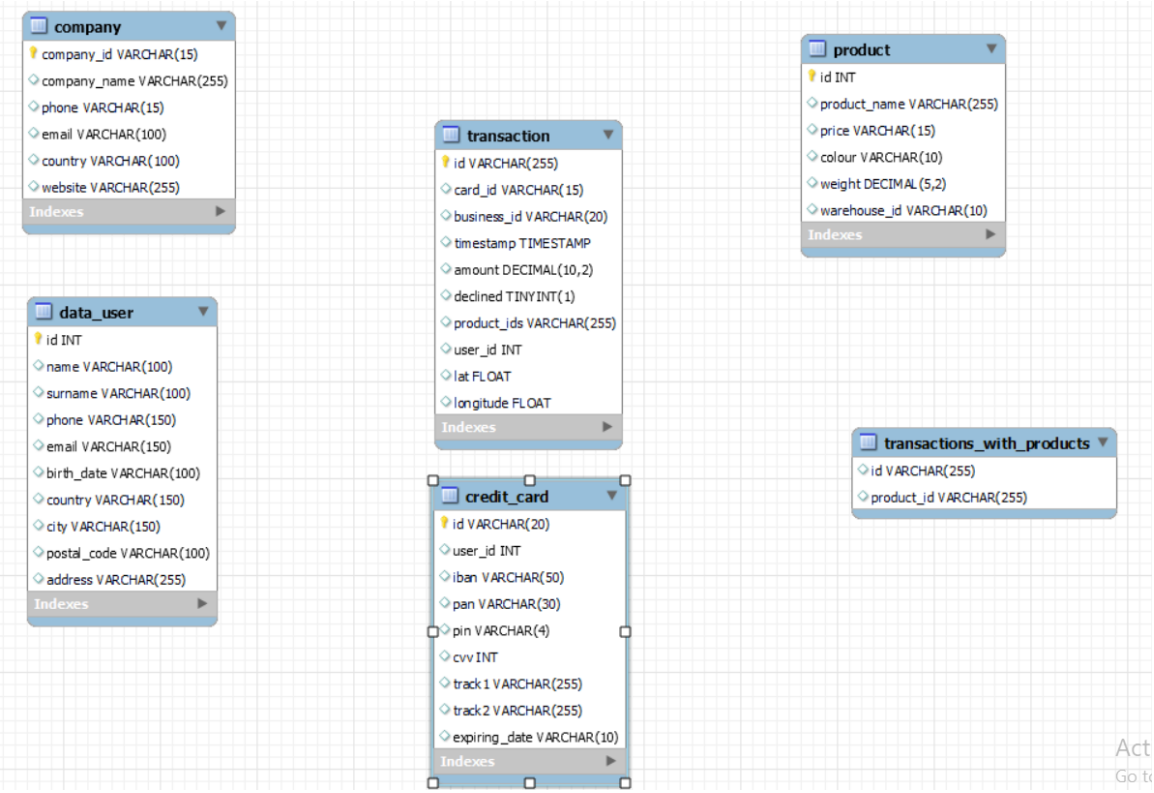
```sql
INSERT INTO transactions_with_products (id, product_id)
SELECT
    id,
    SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', n), ',', -1) AS product_id
FROM
    transaction
JOIN
    (SELECT 1 AS n UNION ALL SELECT 2 UNION ALL SELECT 3 UNION ALL SELECT 4 UNION ALL SELECT 5) AS numbers
ON
    CHAR_LENGTH(product_ids) - CHAR_LENGTH(REPLACE(product_ids, ',', '')) >= n - 1;
```

Comprobamos que el resultado es correcto:

```
SELECT * FROM transactions_with_products;
```

id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
0466A42E-47CF-8D24-FD01-C0B689713128	43
0466A42E-47CF-8D24-FD01-C0B689713128	97
0466A42E-47CF-8D24-FD01-C0B689713128	47
063FBA79-99EC-66FB-29F7-25726D1764A5	5
063FBA79-99EC-66FB-29F7-25726D1764A5	31
063FBA79-99EC-66FB-29F7-25726D1764A5	67
063FBA79-99EC-66FB-29F7-25726D1764A5	47
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	79
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	83
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	31
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43
07A46D48-31A3-7E87-65B9-0DA902AD109F	23

- Ahora nos faltará añadir todas las relaciones de las tablas.



Procedemos a tirar los siguientes comandos:

Primero modificamos las relaciones de la tabla transactions:

transaction.card_id(FK) = credit_card.id(PK)

transaction.business_id(FK) = company.business_id(PK)

transaction.user_id(FK) = data_user.id(PK)

```
#Procedemos a realizar las relaciones entre tablas:
ALTER TABLE transaction
ADD CONSTRAINT fk_card_id FOREIGN KEY (card_id) REFERENCES credit_card(id),
ADD CONSTRAINT fk_business_id FOREIGN KEY (business_id) REFERENCES company(company_id),
ADD CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES data_user(id);
```

A continuacion, he hecho los cambios en la tabla transactions_with_products:

1- A la hora de generar la relación, me ha dado problemas con el index.

2- He generado la relacion de transactions_with_products.id(FK) = transaction.id(PK)

3) He tenido que cambiar el formato de transactions_with_products.product_id a INTEGER para poder añadir la relación.

4) Por último, he añadido la relacion entre transactions_with_products.product_id(FK) = product.id(PK)

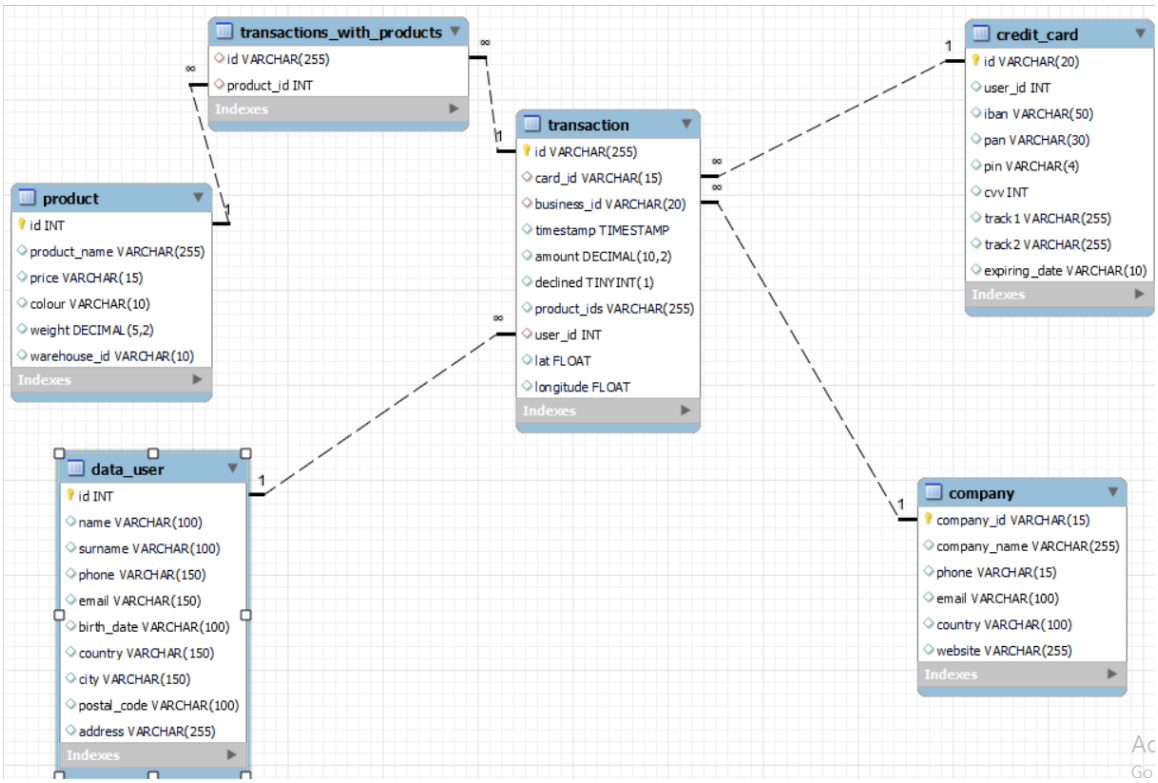
```
#Procedo a crear un index:
CREATE INDEX idx_twp_id ON transactions_with_products(id);

ALTER TABLE transactions_with_products
ADD CONSTRAINT fk_twp_id FOREIGN KEY (id) REFERENCES transaction(id);

ALTER TABLE transactions_with_products
MODIFY product_id INT;

ALTER TABLE transactions_with_products
ADD CONSTRAINT fk_product_id FOREIGN KEY (product_id) REFERENCES product(id);
```

Resultado final:



- Exercici 1

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

```sql
SELECT data_user.id ,data_user.name AS Nombre_usuario, COUNT(*) AS total_transacciones
FROM data_user
JOIN transaction ON transaction.user_id = data_user.id
GROUP BY data_user.id ,data_user.name
HAVING total_transacciones > 30;
```

id	Nombre_usuario	total_transacciones
92	Lynn	39
275	Kenyon	48
272	Hedwig	76
267	Ocean	52

En esta query procedemos a realizar una búsqueda de los nombres de usuario, contando el número de registros que tienen dentro de la tabla transaction. Juntaremos ambas tablas, agruparemos por el id y nombre del usuario, así activamos el count y filtramos con el having a que sean más de 30 las transacciones.

Como resultado, obtenemos que 4 usuarios han realizado mas de 30 transacciones.

- Exercici 2

Mostra la mitjana de la suma de transaccions per IBAN de les targetes de crèdit en la companyia Donec Ltd. utilitzant almenys 2 taules.

```sql
SELECT AVG(transaction.amount) AS media_transacciones, company.company_name
FROM company
JOIN transaction ON transaction.business_id = company.company_id
WHERE company.company_name = "Donec Ltd" #AND transaction.declined = 0
GROUP BY company.company_name;
```

media_transacciones	company_name
203.715000	Donec Ltd

En esta query queremos obtener la media en las transacciones que ha realizado la compañía Donec Ltd.

Procedemos a seleccionar el nombre de la compañía y el Average de amount, juntamos ambas tablas con un JOIN. Y por último agrupamos por compañía.

El resultado nos da 42.82, debido a que es la única transacción que ha realizado aprobada este cliente.

id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
52B1839C-D594-EB3D-4A72-730B1C8B08F4	CcU-2973	b-2242	2021-07-31 23:03:21	364.61	1	53, 59, 5, 41	275	-55.8151	-139.586
5B0EEF86-B8A1-EFAA-5EE1-27E7DC8F54A4	CcU-2973	b-2242	2022-01-06 01:44:48	42.82	0	23, 19, 71	275	-64.1136	85.2491

Nivell 2

- Exercici 1

Quantes targetes estan actives?

```
CREATE TABLE estat_actual_tarjetes (
WITH ranked_transactions AS (
    SELECT
        id,
        card_id,
        business_id,
        timestamp,
        amount,
        declined,
        product_ids,
        user_id,
        lat,
        longitude,
        ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS row_num,
        COUNT(*) OVER (PARTITION BY card_id) AS total_transactions
    FROM
        transaction
```

```sql
SELECT
    card_id,
    CASE
        WHEN total_transactions >= 3 AND SUM(declined) = 3 THEN 'Tarjeta cancelada'
        ELSE 'Tarjeta activa'
    END AS estado_tarjeta
FROM
    ranked_transactions
WHERE
    row_num <= 3
GROUP BY
    card_id, total_transactions
ORDER BY
    card_id);

SELECT *
FROM estat_actual_tarjetes;
```

	card_id	estado_tarjeta
	CcU-2952	Tarjeta activa
	CcU-2959	Tarjeta activa
	CcU-2966	Tarjeta activa
	CcU-2973	Tarjeta activa
	CcU-2980	Tarjeta activa
	CcU-2987	Tarjeta activa
	CcU-2994	Tarjeta activa
	CcU-3001	Tarjeta activa
	CcU-3008	Tarjeta activa
	CcU-3015	Tarjeta activa
	CcU-3022	Tarjeta activa
	CcU-3029	Tarjeta activa
	CcU-3036	Tarjeta activa
	CcU-3043	Tarjeta activa
	CcU-3050	Tarjeta activa
	CcU-3057	Tarjeta activa
	CcU-3064	Tarjeta activa
	CcU-3071	Tarjeta activa
	CcU-3078	Tarjeta activa
	CcU-3085	Tarjeta activa
	CcU-3092	Tarjeta activa

- En esta query procedemos a crear una nueva tabla que se llama estat_actual_tarjetes, en el cual podemos ver si la tarjeta está activa según si las tres últimas transacciones han sido denegadas.

Necesitaremos utilizar las funciones ROW_NUMBER, OVER, PARTITION BY para poder obtener los 3

últimos registros de cada tarjeta filtrado por el timestamp de la transacción que ha sido aprobada.

Una vez tenemos esto filtrado necesitaremos hacer un condición que nos dé la respuesta de sí la tarjeta está activa o no, utilizaremos el CASE.

En el resultado final vemos que TODAS las tarjetas siguen activas, ya que ninguna cumple que haya tenido las 3 últimas transacciones declined.

Nivell 3

- Exercici 1

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

UPDATE: Por lo visto tenia algun problema con la tabla creada y no me hacía el conteo correctamente.

```
SELECT COUNT(product_id), product_name
FROM transactions_with_products, product
WHERE transactions_with_products.product_id = product.id
GROUP BY product_name;
```

COUNT(product_id)	product_name
106	Direwolf Stannis
100	skywalker ewok
68	riverlands north
68	Winterfell
66	Direwolf riverlands the
65	Tarly Stark
65	duel
62	Tully

En esta query procedemos a ver cuantas veces se ha vendido cada producto. Para obtener este resultado procedemos a buscar por el nombre de producto, y hacemos un count de todos los registros dentro de la tabla transactions_with_products y lo juntamos con la tabla product para poder mostrar el nombre del producto

Podemos ver que el producto más vendido ha sido Direwolf Stannis con 106 ventas.