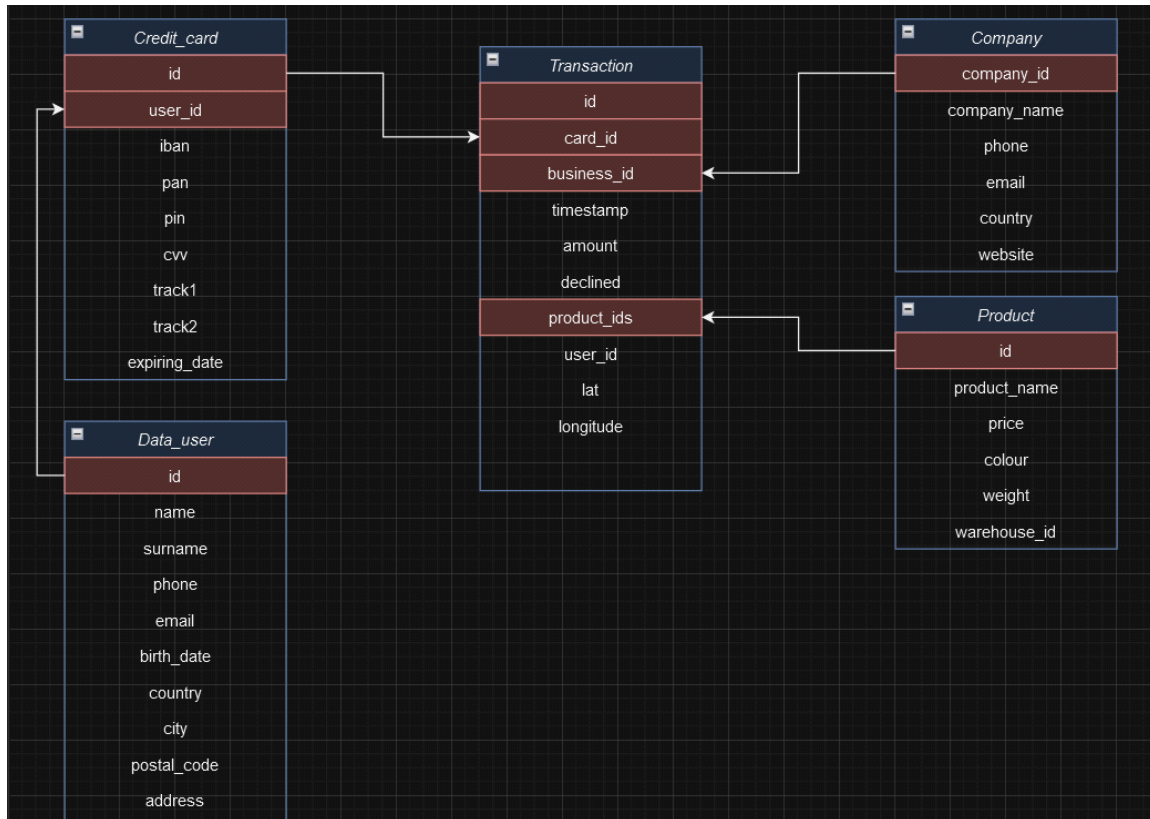


Nivell 1

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:

Generamos el modelo



A continuació 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](#)

```
CREATE TABLE product (
  id INT PRIMARY KEY,
  product_name VARCHAR(255),
  price DECIMAL(10, 2),
  colour VARCHAR(10),
  weight DECIMAL(5, 2),
  warehouse_id VARCHAR(10)
);
```

- Al intentar añadir los datos me daba problemas, debido a que el campo price tenía el símbolo \$, y eso nos impedirá en el futuro poder operar con los precios, procedemos a borrar el símbolo. Primero cambio el formato del campo para que me dejé introducir los datos sin problemas.

```
ALTER TABLE product
MODIFY COLUMN price VARCHAR(15);
```

- Una vez ya hemos añadido toda la información, procedemos a borrar el símbolo y cambiarle el formato al campo, ahora pasará a ser decimal.

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

4) Creamos la tabla [transaction](#)

```
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`

```
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 búsquedas por los productos comprados individualmente, ya que en `products_id`, salen todos los productos comprados por cada transacción.

```
CREATE TABLE transactions_with_products (
    id VARCHAR(255),
    card_id VARCHAR(255),
    business_id VARCHAR(255),
    timestamp TIMESTAMP,
    amount FLOAT,
    declined INT,
    product_id VARCHAR(255),
    user_id INT,
    lat FLOAT,
    longitude FLOAT
);
```

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

```
INSERT INTO transactions_with_products (id, card_id, business_id, timestamp, amount, declined, product_id, user_id, lat, longitude)
SELECT
    id,
    card_id,
    business_id,
    timestamp,
    amount,
    declined,
    SUBSTRING_INDEX(SUBSTRING_INDEX(product_ids, ',', n), ',', -1) AS product_id,
    user_id,
    lat,
    longitude
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;
```

- Exercici 1

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

```
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.

```
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
5B0EEF86-88A1-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_tarjetas (  
> 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  
  
    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_tarjetas;
```

	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_tarjetas, 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.

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

	product_name	product_id	COUNT(*)
►	Winterfell	67	44
	Tarly Stark	2	44
	jinn Winterfell	97	40
	Direwolf Stannis	1	39
	palpatine chewbacca	13	38
	duel	43	36
	riverlands north	23	36
	Tully	47	36
	Tully Dorne	71	36
	Direwolf riverlands the	79	35
	duel tourney	83	35
	Tully maester Tarly	29	35
	skywalker ewok sith	17	35

En esta query procedemos a ver cuantas veces se ha vendido cada producto. Para obtener este resultado procedemos a buscar por el ID y 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 los 2 productos mas vendidos han realizado 44 ventas cada uno.