EXERCICIS SPRINT 4

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.

Voy a crear una base de datos llamada *transactionsBD*.

```
5 •  CREATE DATABASE TransactionsBD;
6 •  USE TransactionsBD;
```

Despues de ver los archivos CSV voy a realizar 4 tablas, a saber, tabla companies, tabla credit_cards, tabla transactions y la tabla users que será la unificación de los archivos csv users_ca, users_uk y users_usa. Son tres archivos que separan la información por países, pero ya tienen un campo country, por lo tanto puedo ponerlos en la misma tabla y con querys ya podre separarlos por países.

```
 8     -- Creamos la tabla `users`
 9 •   CREATE TABLE users (
10         id INT PRIMARY KEY,
11         name VARCHAR(50),
12         surname VARCHAR(50),
13         phone VARCHAR(20),
14         email VARCHAR(100),
15         birth_date VARCHAR(20),
16         country VARCHAR(50),
17         city VARCHAR(50),
18         postal_code VARCHAR(20),
19         address VARCHAR(100)
20     );
```

```
✓  10  13:34:09  CREATE TABLE users (  id INT PRIMARY KEY,  name VARCHAR(50),  surname VARCHAR(50),  ...  0 row(s) affected
```

```
-- Creamos la tabla `credit_cards`
CREATE TABLE credit_cards (
    id VARCHAR(20) PRIMARY KEY,
    user_id INT,
    iban VARCHAR(34),
    pan VARCHAR(19),
    pin VARCHAR(10),
    cvv VARCHAR(4),
    track1 VARCHAR(100),
    track2 VARCHAR(100),
    expiring_date VARCHAR(20),
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

```
✓  11  13:38:14  CREATE TABLE credit_cards (  id VARCHAR(20) PRIMARY KEY,  user_id INT,  iban VARCHAR(3...  0 row(s) affected
```

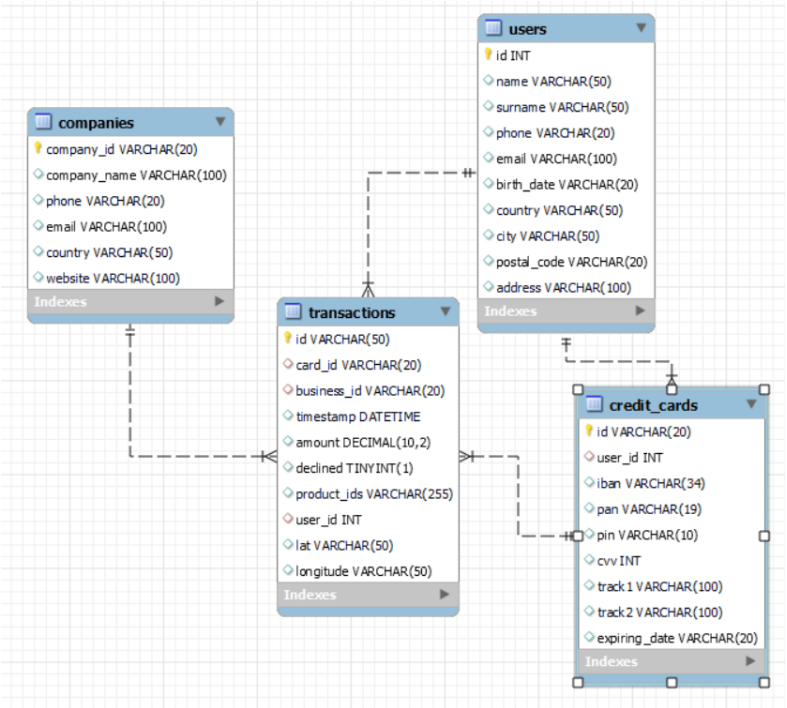
```sql
-- Creamos la tabla `companies`
CREATE TABLE companies (
    company_id VARCHAR(20) PRIMARY KEY,
    company_name VARCHAR(100),
    phone VARCHAR(20),
    email VARCHAR(100),
    country VARCHAR(50),
    website VARCHAR(100)
);
```

```
12  13:39:36  CREATE TABLE companies (  company_id VARCHAR(20) PRIMARY KEY,  company_name VARCH...  0 row(s) affected
```

```sql
-- Creamos la tabla `transactions`
CREATE TABLE transactions (
    id VARCHAR(50) PRIMARY KEY,
    card_id VARCHAR(20),
    business_id VARCHAR(20),
    timestamp DATETIME,
    amount DECIMAL(10,2),
    declined BOOLEAN,
    product_ids VARCHAR(255),
    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)
);
```

```
13  13:40:55  CREATE TABLE transactions (  id VARCHAR(50) PRIMARY KEY,  card_id VARCHAR(20),  busine...  0 row(s) affected
```

El diagrama de como quedara la base de datos, con sus relaciones y tipo de datos por columnas, es el siguiente:



Vamos a introducir los datos a partir de los archivos .csv proporcionados.

Tabla users:

```
LOAD DATA LOCAL INFILE 'C:\\Program Files\\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 LOCAL INFILE 'C:\\Program Files\\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 LOCAL INFILE 'C:\\Program Files\\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;
```

Tabla credit_cards:

```
LOAD DATA LOCAL INFILE 'C:\\Program Files\\MySQL\\MySQL Server 8.0\\uploads\\credit_cards.csv'
INTO TABLE credit_cards
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

Tabla companies:

```
LOAD DATA LOCAL INFILE 'C:\\Program Files\\MySQL\\MySQL Server 8.0\\uploads\\companies.csv'
INTO TABLE companies
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

Tabla transactions:

```
LOAD DATA LOCAL INFILE 'C:\\Program Files\\MySQL\\MySQL Server 8.0\\uploads\\transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

Comprobamos la información introducida

```
65 •  SELECT *
66    FROM companies;
67
```

company_id	company_name	phone	email	country	website
b-2222	Ac Fermentum Incorporated	06 85 56 52 33	donec.porttitor.tellus@yahoo.net	Germany	https://instagram.com/site
b-2226	Magna A Neque Industries	04 14 44 64 62	risus.donec.nibh@icloud.org	Australia	https://whatsapp.com/group/9
b-2230	Fusce Corp.	08 14 97 58 85	risus@protonmail.edu	United States	https://pinterest.com/sub/cars
b-2234	Convallis In Incorporated	06 66 57 29 50	mauris.ut@aol.couk	Germany	https://cnn.com/user/110

```
31  13:00:53  SELECT * FROM companies LIMIT 0, 1000                100 row(s) returned
```

```
68 •  SELECT *
69    FROM credit_cards;
70
```

id	user_id	iban	pan	pin	cvv	track1	track2
CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%B8383712448554646^WovsxejDpwiev^8604...	%B7653863056044187=80071
CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%B4621311609958661^UftuyfsSeimxn^06106...	%B4149568437843501=51071
CcU-2952	273	BG45IVQL52710525608255	4556 453 55 5287	4598	438	%B2183285104307501^CddyytcUxwfdq^5907...	%B6778580257827162=69068
CcU-2959	272	CR7242477244335841535	372461377349375	3583	667	%B7281111956795320^XocddjBckecd^09016...	%B4246154489281853=28052

```
131  13:35:43  SELECT * FROM credit_cards LIMIT 0, 1000              275 row(s) returned
```

```
71 •  SELECT *
72    FROM transactions;
73
```

id	card_id	business_id	timestamp	amount	declined	product_ids	user_id	lat	longitude
02C6201E-D90A-1859-B4EE-88D2986D3B02	CcU-2938	b-2362	2021-08-28 23:42:24	466.92	0	71, 1, 19	92	81.9184589824	-12.5275561984
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	CcU-4219	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.9694885888	-117.525183590
063FBA79-99EC-66FB-29F7-25726D1764A5	CcU-2987	b-2230	2022-01-06 21:25:27	92.61	0	47, 67, 31, 5	275	-81.222680576	-129.049879552
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	CcU-3743	b-2618	2022-01-26 02:07:14	394.18	0	89, 83, 79	265	-34.3593055232	-100.555928064

```
132  13:37:27  SELECT * FROM transactions LIMIT 0, 1000             587 row(s) returned
```

```
74 •  SELECT *
75    FROM users;
```

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	Nov 17, 1985	United States	Lowell	73544	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	Aug 23, 1992	United States	Des Moines	59464	903 Sit Ave
3	Ciaran	Harrison	(522) 598-1365	(718) 257-2412	Apr 29, 1998	United States	Columbus	56518	736-2063 Tellus St.
4	Howard	Stafford	1-411-740-3269	ornare.egestas@icloud.edu	Feb 18, 1989	United States	Kailua	77417	Ap #545-2244 Erat. Rd.

```
133  13:39:41  SELECT * FROM users LIMIT 0, 1000                   275 row(s) returned
```

Ya tenemos todo a punto para hacer los ejercicios

EXERCICI 1

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

```sql
SELECT u.name, u.surname
FROM users u
WHERE u.id IN (
    SELECT t.user_id
    FROM transactions t
    GROUP BY t.user_id
    HAVING COUNT(*) > 30
);
```

Result Grid | Filter Rows:

name	surname
Lynn	Riddle
Ocean	Nelson
Hedwig	Gilbert
Kenyon	Hartman

1 13:46:25 SELECT u.name, u.surname FROM users u WHERE u.id IN (SELECT t.user_id FROM transactions t... 4 row(s) returned

EXERCICI 2

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

```sql
SELECT cc.iban, ROUND(AVG(t.amount), 2) as mitjana_amount
FROM credit_cards cc
JOIN transactions t ON cc.id = t.card_id
JOIN companies c ON t.business_id = c.company_id
WHERE c.company_name = 'Donec Ltd'
GROUP BY cc.iban;

```

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

iban	mitjana_amount
PT87806228135092429456346	203.72

4 20:10:23 SELECT cc.iban, ROUND(AVG(t.amount), 2) as mitjana_amount FROM credit_cards cc JOIN transaction... 1 row(s) returned

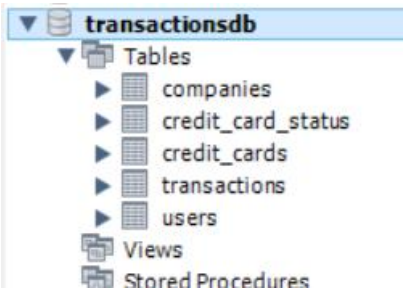
NIVELL 2

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

Vamos a crear la tabla:

```sql
CREATE TABLE credit_card_status (
    card_id VARCHAR(20) PRIMARY KEY,
    status VARCHAR(50) NOT NULL
);
```

```
5  20:29:11  CREATE TABLE credit_card_status (  card_id VARCHAR(20) PRIMARY KEY,  status VARCHAR(50) ...  0 row(s) affected
```



Vamos a llenarla de datos:

```sql
INSERT INTO credit_card_status (card_id, status)
SELECT c.id AS card_id,
    CASE
        WHEN COUNT(t.id) >= 3 AND SUM(t.declined) = 3 THEN 'Bloquejada'
        ELSE 'Activa'
    END AS status
FROM credit_cards c
LEFT JOIN transactions t ON c.id = t.card_id
GROUP BY c.id;
```

```
15  20:42:12  INSERT INTO credit_card_status (card_id, status) SELECT c.id AS card_id,  CASE  WHEN COU...  275 row(s) affected Records: 275  Duplicates: 0  Warnings: 0
```

Vamos a ver algunos valores introducidos:

```sql
SELECT *
FROM credit_card_status;
```

Result Grid | Filter Rows:

card_id	status
CcU-2938	Activa
CcU-2945	Activa
CcU-2952	Activa
CcU-2959	Activa
CcU-2966	Activa

```
17  20:46:07  SELECT * FROM credit_card_status LIMIT 0, 1000  275 row(s) returned
```

EXERCICI 1

Quantes targetes estan actives?

```sql
SELECT DISTINCT COUNT(card_id) AS TarjetasActivas
FROM credit_card_status
WHERE status='Activa';
```

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

TarjetasActivas
275

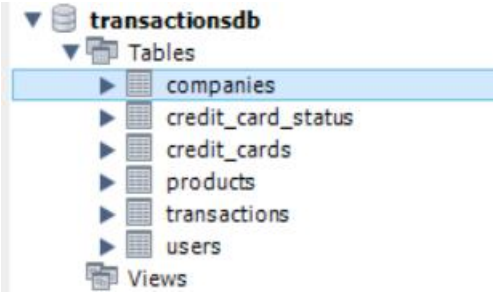
NIVELL 3

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:

Creamos la tabla:

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

21 21:07:21 CREATE TABLE products (id INT PRIMARY KEY, product_name VARCHAR(100), price DECI... 0 row(s) affected



Ahora lo rellenamos de datos con el archivo csv proporcionado

```sql
LOAD DATA LOCAL INFILE 'C:\\Program Files\\MySQL\\MySQL Server 8.0\\uploads\\products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

Comprobamos los datos:

```sql
SELECT *
FROM products;
```

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
3	duel tourney Lannister	$171.13	#d8d8d8	1.50	WH-2
4	warden south duel	$71.89	#111111	3.00	WH-1
5	skywalker ewok	$171.22	#dbdbdb	3.20	WH-0
6	dooku solo	$136.60	#c4c4c4	0.80	WH--1

products 12

37 21:29:36 SELECT * FROM products LIMIT 0, 1000 100 row(s) returned

Creamos una tabla entremedia para relacionar tabla products con tabla transactions

```sql
CREATE TABLE transaction_products (
    transaction_id VARCHAR(50),
    product_id INT,
    FOREIGN KEY (transaction_id) REFERENCES transactions(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

1 10:10:05 CREATE TABLE transaction_products (transaction_id VARCHAR(50), product_id INT, FOREIG... 0 row(s) affected

Insertamos los datos a la nueva tabla transaction_products. Los datos de product_id los convertimos en una array list y estos los convertimos en una Json_table, asi cada elemento de la array se convierte en una fila.

```sql
INSERT INTO transaction_products (transaction_id, product_id)
SELECT t.id, p.value
FROM transactions t
CROSS JOIN JSON_TABLE(
    CONCAT('["', REPLACE(t.product_ids, ', ', '","'), '"]'),
    '$[*]' COLUMNS (value VARCHAR(255) PATH '$')
) AS p;
```

2 10:14:12 INSERT INTO transaction_products (transaction_id, product_id) SELECT t.id, p.value FROM transactio... 1457 row(s) affected Records: 1457 Duplicates: 0 Warnings: 0

Comprobamos los datos introducidos

```
165 •   SELECT *
166     FROM transaction_products;
167
```

Result Grid | Filter Rows: | Export:

transaction_id	product_id
09DE92CE-6F27-2BB7-13B5-9385B2B3B8E2	7
0A476ED9-0C13-1962-F87B-D3563924B539	29
0A476ED9-0C13-1962-F87B-D3563924B539	41
0A476ED9-0C13-1962-F87B-D3563924B539	11
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	19
0BEB80B7-9D66-1707-CE4B-9DC7E71914B5	41

1 10:42:45 SELECT * FROM transaction_products LIMIT 0, 1000 1000 row(s) returned

EXERCICI 1

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

```
170 •   SELECT p.id, p.product_name, p.colour, COUNT(tp.product_id) AS sales_count
171     FROM products p
172     LEFT JOIN transaction_products tp ON p.id = tp.product_id
173     GROUP BY p.id, p.product_name, p.colour
174     ORDER BY sales_count DESC;
```

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

id	product_name	colour	sales_count
23	riverlands north	#545454	68
67	Winterfell	#1c1c1c	68
79	Direwolf riverlands the	#b2b2b2	66
2	Tarly Stark	#919191	65
43	duel	#5b5b5b	65
47	Tully	#919191	62

6 10:53:21 SELECT p.id, p.product_name, p.colour, COUNT(tp.product_id) AS sales_count FROM products p LEF... 100 row(s) returned