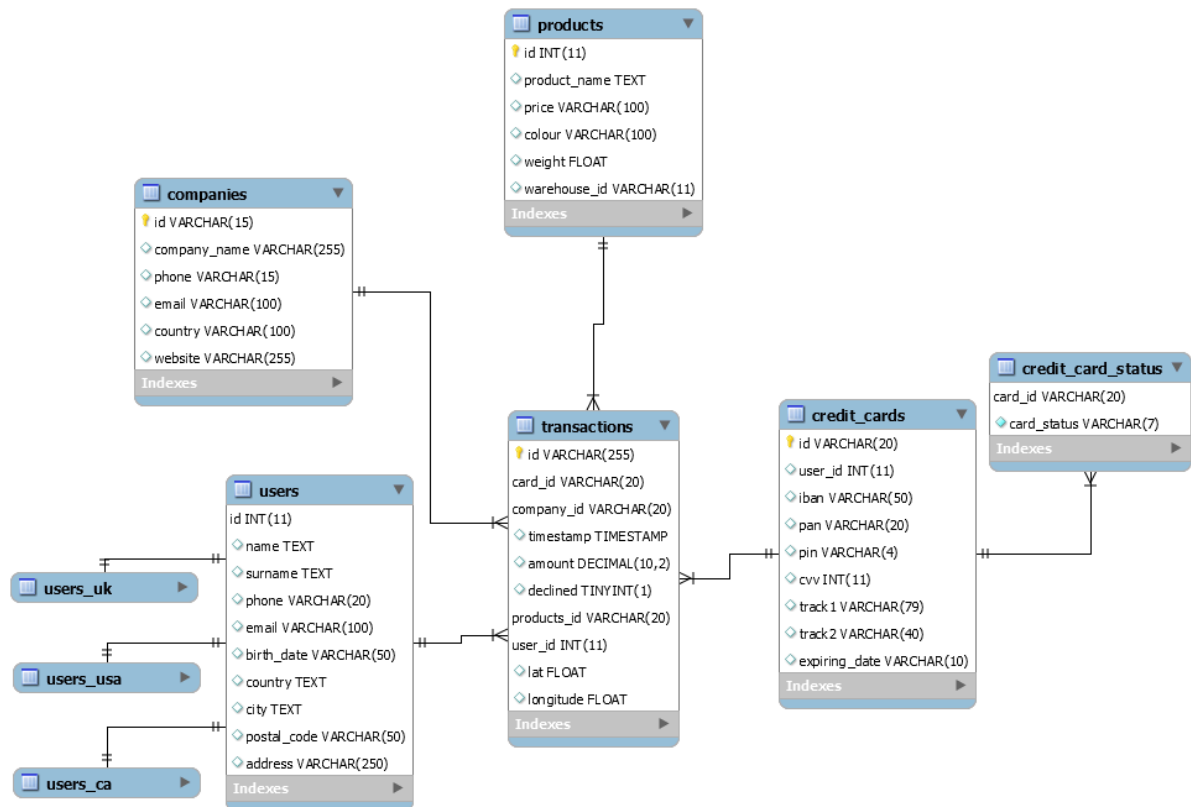


Nivell 1

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:



```
1 CREATE DATABASE transas;
2
3
4 -- Creemos la tabla users_ca
5 CREATE TABLE IF NOT EXISTS users_ca (
6   id int(11) PRIMARY KEY,
7   name text,
8   surname text,
9   phone varchar(20) DEFAULT NULL,
10  email varchar(100) DEFAULT NULL,
11  birth_date varchar(50) DEFAULT NULL,
12  country text,
13  city text,
14  postal_code varchar(50) DEFAULT NULL,
15  address varchar(250) DEFAULT NULL
16 );
17
18 -- Creemos la tabla users_uk
19 CREATE TABLE IF NOT EXISTS users_uk (
20   id int(11) PRIMARY KEY,
21   name text,
22   surname text,
23   phone varchar(20) DEFAULT NULL,
24   email varchar(100) DEFAULT NULL,
25   birth_date varchar(50) DEFAULT NULL,
26   country text,
27   city text,
28   postal_code varchar(50) DEFAULT NULL,
29   address varchar(250) DEFAULT NULL
30 );
```

```
61
62 -- Creemos la tabla credit_cards
63 CREATE TABLE IF NOT EXISTS credit_cards (
64   id VARCHAR(20) PRIMARY KEY NOT NULL,
65   user_id INT(11) NULL DEFAULT NULL,
66   iban VARCHAR(50) NULL DEFAULT NULL,
67   pan VARCHAR(20) NULL DEFAULT NULL,
68   pin VARCHAR(4) NULL DEFAULT NULL,
69   cvv INT(11) NULL DEFAULT NULL,
70   track1 VARCHAR(79) NULL DEFAULT NULL,
71   track2 VARCHAR(40) NULL DEFAULT NULL,
72   expiring_date VARCHAR(10) NULL DEFAULT NULL
73 );
74
75 -- Creemos la tabla companies
76 CREATE TABLE IF NOT EXISTS companies (
77   id VARCHAR(15) PRIMARY KEY NOT NULL,
78   company_name VARCHAR(255) NULL DEFAULT NULL,
79   phone VARCHAR(15) NULL DEFAULT NULL,
80   email VARCHAR(100) NULL DEFAULT NULL,
81   country VARCHAR(100) NULL DEFAULT NULL,
82   website VARCHAR(255) NULL DEFAULT NULL
83 );
```

```

31
32 -- Creemos la tabla users_usa
33 CREATE TABLE IF NOT EXISTS users_usa (
34     id int(11) PRIMARY KEY,
35     name text,
36     surname text,
37     phone varchar(20) DEFAULT NULL,
38     email varchar(100) DEFAULT NULL,
39     birth_date varchar(50) DEFAULT NULL,
40     country text,
41     city text,
42     postal_code varchar(50) DEFAULT NULL,
43     address varchar(250) DEFAULT NULL
44 );
45
46
47 -- Creemos la tabla transaction
48 CREATE TABLE IF NOT EXISTS transactions (
49     id VARCHAR(255) PRIMARY KEY NOT NULL,
50     card_id VARCHAR(20) REFERENCES credit_cards(id),
51     business_id VARCHAR(20),
52     timestamp TIMESTAMP,
53     amount DECIMAL(10, 2),
54     declined BOOLEAN,
55     products_id VARCHAR(20) REFERENCES products(id),
56     user_id INT REFERENCES users(id),
57     lat FLOAT,
58     longitude FLOAT
59 );
60

```

```

-- Creemos la tabla users
CREATE TABLE IF NOT EXISTS users (
    id int(11) PRIMARY KEY,
    name text,
    surname text,
    phone varchar(20) DEFAULT NULL,
    email varchar(100) DEFAULT NULL,
    birth_date varchar(50) DEFAULT NULL,
    country text,
    city text,
    postal_code varchar(50) DEFAULT NULL,
    address varchar(250) DEFAULT NULL
);

-- añadimos los registros de la tabla users_ca a la tabla users
INSERT INTO users
SELECT * FROM users_ca;

-- añadimos los registros de la tabla users_usa a la tabla users
INSERT INTO users
SELECT * FROM users_usa;

-- añadimos los registros de la tabla users_uk a la tabla users
INSERT INTO users
SELECT * FROM users_uk;

```

En las imágenes, se muestra la creación de las tablas de nuestra base de datos. Como se puede ver en la primera imagen tenemos el diagrama final de nuestra base de datos, en el cuál se muestra todas las tablas que lo componen y sus relaciones de cardinalidad entre ellas. También, se ha creado una tabla users, la cuál contiene todo los datos de users_ca, users_usa y users_uk. La creación de esta tabla es debido a que las tres tablas users_ca, users_usa y users_uk tienen la misma estructura. Se ha creado una tabla users con la misma estructura, y se han cargado los datos de las tres tablas en users.

EXERCICI 1

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

```

112
113 #EXERCICI 1
114 #Realitza una subconsulta que mostri tots els usuaris amb més de 30 transaccions utilitzant almenys 2 taules.
115
116 SELECT id,name
117 FROM users
118 WHERE id IN (SELECT user_id
119             FROM transactions
120             GROUP BY user_id
121             HAVING COUNT(amount) > 30);
122

```

id	name
92	Lynn
267	Ocean
272	Hedwig
275	Kenyon
NULL	NULL

Los usuarios con más de 30 transacciones son Lynn, Ocean, Hedwig y Kenyon.

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.

```
123 #EXERCICI 2
124 #Mostra la mitjana de la suma de transaccions per IBAN de les targetes de crèdit
125 #en la companyia Donec Ltd. utilitzant almenys 2 taules.
126
127 -- cambiamos el nombre del campo business_id por company_id para que sea más reconocible.
128 • ALTER TABLE transactions
129   CHANGE business_id company_id varchar(20);
130
131
132 • SELECT AVG(t.amount) as MediaAmount
133   FROM transactions t
134   JOIN companies c
135   ON c.id = t.company_id
136   JOIN (SELECT cc.iban, SUM(t.amount) as Total_amount
137         FROM transactions t
138         JOIN credit_cards cc
139         ON t.card_id = cc.id
140         GROUP BY cc.iban) SumTotal
141   WHERE company_name = 'Donec Ltd'
142   GROUP BY company_id;
143
144 ...
```

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

MediaAmount
203.715000

La media de la suma de las transacciones por IBAN de 'Donec Ltd' es de 203,715.

Hemos usado una tabla derivada en el JOIN para obtener el resultado.

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:

```
145
146 #NIVELL 2
147 #Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les últimes tres transaccions
148 #van ser declinades i genera la següent consulta:
149 #EXERCICI 1
150 #Quantes targetes estan actives?
151
152 -- Creamos la tabla credit_card_status
153 • CREATE TABLE IF NOT EXISTS credit_card_status AS
154
155   SELECT card_id,
156         CASE
157           WHEN SUM(declined)>= 3 THEN 'Blocked'
158           ELSE 'Active'
159         END AS card_status
160   FROM (SELECT card_id, timestamp, declined,
161             ROW_NUMBER() OVER (PARTITION BY card_id
162                               ORDER BY timestamp DESC) AS num_registro
163         FROM transactions) AS Registers
164   WHERE num_registro<=3
165   GROUP BY card_id;
```

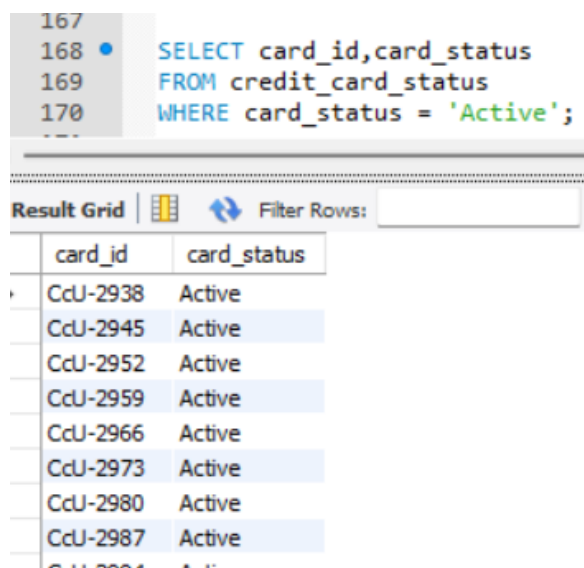
La foto anterior, muestra la creación de la tabla **credit_card_status**, que nos indica que tarjetas de crédito están activas y cuáles no.

Para poder crearlo se ha realizado un contador que cuenta las últimas tres transacciones de cada tarjeta de crédito si es que se han realizado 3 transacciones. Después se ha evaluado si la tarjeta de crédito estaba bloqueada o no según si sus 3 últimas transacciones han sido declinadas.

El resultado, es una tabla, con dos columnas, una con la **card_id** que nos indica la id de la tarjeta de crédito y otra con **card_status** que nos indica si la tarjeta está 'Blocked' o 'Active'.

EXERCICI 1

Quantes targetes estan actives?



The screenshot shows a SQL query in a text editor and its corresponding result grid. The query is: `SELECT card_id, card_status FROM credit_card_status WHERE card_status = 'Active';` The result grid displays a table with two columns: **card_id** and **card_status**. All entries in the **card_status** column are 'Active'.

card_id	card_status
CcU-2938	Active
CcU-2945	Active
CcU-2952	Active
CcU-2959	Active
CcU-2966	Active
CcU-2973	Active
CcU-2980	Active
CcU-2987	Active

Según el resultado de nuestra query, ninguna tarjeta ha sido declinada 3 veces en sus últimas 3 transacciones. Por lo tanto, todas las tarjetas están activas.

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:

```
#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:

#EXERCICI 1
#Necessitem conèixer el nombre de vegades que s'ha venut cada producte.

CREATE TABLE IF NOT EXISTS products (
  id INT(11) PRIMARY KEY NOT NULL AUTO_INCREMENT,
  product_name TEXT NULL DEFAULT NULL,
  price VARCHAR(100) NULL DEFAULT NULL,
  colour VARCHAR(100) NULL DEFAULT NULL,
  weight FLOAT NULL DEFAULT NULL,
  warehouse_id VARCHAR(11) NULL DEFAULT NULL
);
```

Se ha creado la tabla **products** para poder relacionarla con la tabla **transactions**, ya que contiene el campo **products_id**.

La tabla **products** almacena los datos de los productos, pero si nos fijamos, hay nombres de productos iguales, con id's diferentes, por lo tanto se tiene que tener en cuenta cuando agrupamos, porque puede dar valores diferentes.

EXERCICI 1

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

```
191
192 • SELECT p.product_name, COUNT(p.id) AS Num_Sales
193 FROM transactions t
194 JOIN products p
195 ON FIND_IN_SET(p.id, REPLACE(t.products_id, ' ', '')) > 0
196 GROUP BY p.product_name;
197
198 #Se tiene que ordenar por product_name, porque hay varios productos que se llaman igual pero tienen diferentes ID's.
199
```



The screenshot shows a database interface with a query editor and a result grid. The query is a SQL statement that joins the **transactions** table with the **products** table using a **FIND_IN_SET** function to match product IDs. The result grid displays the following data:

product_name	Num_Sales
Direwolf Stannis	106
dooku solo	49
Tully Dorne	54
duel	65
Tully	62
jinn Winterfell	61
skywalker ewok	100
Lannister	47
Winterfell	68
Direwolf riverlands the	66
duel tourney	57
riverlands north	68
north of Casterly	54
Karstark Dorne	48

Analizando la tabla **transactions**, vemos que el campo 'products_id' contiene más de una id por campo, por lo tanto, se tiene que separar cada id en una fila para poder contar cuántos productos se han vendido. Se ha usado el **FIND_IN_SET** para poder comprobar que 'id' de la tabla **products** coinciden con los id de 'products_id' de la tabla **transactions**.

Lo que se hace es lo siguiente:

- Se hace un **REPLACE** para que elimine los espacios en blanco del campo 'products_id', ya que los id están separados por comas.

- Después el **FIND_IN_SET** devuelve la posición del valor si se encuentra en la cadena, o 0 si no se encuentra. Por lo tanto, esta parte de la expresión verifica si el id del producto en la tabla **products** está presente en la lista de 'products_id' de la tabla **transactions**.

- Por último si ese valor es más grande que 0, significa que la id de la tabla **products** es verdadera y se encuentra en el campo 'product_id' de la tabla **transactions**.

- Se usa en el **ON** del **JOIN** pero en vez de una igualdad, se está utilizando una comparación.