

# Tasca S4.01. Creació de Base de Dades

Hecho por: Yatmelis Freites

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 GeekStore;
2 • USE GeekStore;
3
4 • CREATE TABLE companies (
5     company_id VARCHAR(100) PRIMARY KEY,
6     company_name VARCHAR(50),
7     phone VARCHAR(20),
8     email VARCHAR(50),
9     country VARCHAR(50),
10    website VARCHAR(100)
11 );
12
13 • CREATE TABLE credit_cards (
14     id VARCHAR(20) PRIMARY KEY,
15     user_id INT,
16     iban VARCHAR(34) NOT NULL UNIQUE,
17     pan VARCHAR(25) NOT NULL,
18     pin INT(4) NOT NULL,
19     cvv INT(3) NOT NULL,
20     track1 VARCHAR(100) ,
21     track2 VARCHAR(100),
22     expiring_date VARCHAR(10) NOT NULL
23 );
```

Primero se declara **CREATE DATABASE** para generar mi base de datos, indicando el nombre a continuación y posteriormente **USE** para indicar que todas las acciones a continuación se realizarán para modificar esta base de datos en específico.

Luego con **CREATE TABLE** escribo el nombre de la tabla a crear, en primer lugar **companies** y entre paréntesis se insertan los campos que se encuentran en el archivo .csv, luego de haberlos analizado para determinar el tipo de dato que contiene, y al crear el campo **company\_id**, se escribe a continuación **PRIMARY KEY** para indicar que es la clave primaria de esta tabla.

Para la tabla **credit\_cards** se escribe la declaración **CREATE TABLE** y posteriormente se introducen los nombres del archivo .csv y a continuación se indica que **id** es la **PRIMARY KEY**, y que el campo **iban** es **NOT NULL** para evitar campos vacíos en caso de que existan y se indica que esta es una **clave UNIQUE**, es decir que solo existe un iban para cada tarjeta de crédito, este dato no se repite, y los campos **pan**, **pin**, **cvv** son igualmente **NOT NULL**, al igual que el campo **expiring\_date** porque toda tarjeta posee una fecha de vencimiento

```

24 • ○ CREATE TABLE products(
25     id VARCHAR(30) PRIMARY KEY ,
26     product_name VARCHAR(20),
27     price VARCHAR(20),
28     colour VARCHAR(7),
29     weight FLOAT,
30     warehouse_id VARCHAR(7)
31 );
32
33 • ○ CREATE TABLE users (
34     id INT PRIMARY KEY,
35     name VARCHAR(30),
36     surname VARCHAR(30),
37     phone VARCHAR(20),
38     email VARCHAR(50),
39     birth_date VARCHAR(25) NOT NULL,
40     country VARCHAR(50),
41     city VARCHAR(50),
42     postal_code VARCHAR (15),
43     address VARCHAR(50)
44 );
45
46

```

Para la tabla **products** se escribe la declaración **CREATE TABLE** y posteriormente se introducen los nombres del archivo .csv y a continuación se indica que **id** es la **PRIMARY KEY**, se completan los tipos de datos para cada campo (**VARCHAR**) según correspondan en el archivo, en este caso solo **weight** posee un tipo de dato **FLOAT** porque el peso de los productos puede ser de tipo decimal.

Para la tabla **users** se escribe la declaración **CREATE TABLE** y posteriormente se introducen los nombres del archivo .csv y a continuación se indica que **id** es la **PRIMARY KEY**, y que el campo **birth\_date** es **NOT NULL**, porque toda persona tiene una una fecha de nacimiento, y todos los campos a excepción de **id** son de tipo **VARCHAR**.

```

47 ● CREATE TABLE transactions (
48     id VARCHAR(100) PRIMARY KEY,
49     card_id VARCHAR(20),
50     business_id VARCHAR(100),
51     timestamp TIMESTAMP,
52     amount DECIMAL(10, 2),
53     declined TINYINT(1),
54     product_ids VARCHAR(30),
55     user_id INT,
56     lat FLOAT,
57     longitude FLOAT,
58     FOREIGN KEY (card_id) REFERENCES credit_cards(id),
59     FOREIGN KEY (business_id) REFERENCES companies(company_id),
60     FOREIGN KEY (user_id) REFERENCES users(id)
61 );

```

Por último, de la tabla **transactions** se escribe la declaración **CREATE TABLE** y posteriormente se introducen los nombres del archivo .csv y a continuación se indica que **id** es la **PRIMARY KEY** de tipo **VARCHAR**, y que los campos **card\_id** y **business\_id** son de tipo **VARCHAR**. El campo **timestamp** es de tipo **TIMESTAMP**, **amount** de tipo **DECIMAL**, indicando entre paréntesis el número de dígitos aceptados, y luego de la coma el número de decimales para que sea más fácil de leer. Para el campo **declined** se indica que es de tipo **TINYINT** (ya que con 1 o 0 se indica si la transacción fue aprobada o rechazada), **product\_ids** es de tipo **VARCHAR**, **user\_id** de tipo **INT** ya que se indica con números y **lat** y **long** son **FLOATS** porque son de tipo decimal.

Se ha identificado que esta es la tabla de hechos y por lo tanto se agregan todas las **FOREIGN KEYS**, referenciando a sus respectivos campos en las tablas de dimensiones.

Cabe resaltar que se complicaba la importación de los datos de **transactions** de manera correcta dado que los datos de la columna **product\_ids** eran diferentes a los de la tabla **product(id)**, por lo tanto no existe como clave foránea en esta tabla

## Nivell 1

### Exercici 1

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

```

4
5 • SELECT u.id, u.name, u.surname, u.email
6 FROM geekstore.users u
7 WHERE u.id IN (
8     SELECT t.user_id
9     FROM geekstore.transactions t
10    GROUP BY t.user_id
11    HAVING COUNT(t.user_id) > 30
12 );

```

id	name	surname	email
92	Lynn	Riddle	vitae.aliquet@outlook.edu
267	Ocean	Nelson	aenean@yahoo.com
272	Hedwig	Gilbert	sem.eget@idcloud.edu
275	Kenyon	Hartman	convallis.ante.lectus@yahoo.com
* NULL	NULL	NULL	NULL

Se escribe **SELECT** del **id**, **name**, **surname** e **email** de la tabla **users** y se usa la letra **u** como alias y se realiza la subconsulta utilizando un **WHERE** para filtrar **id**.

Dentro de la subconsulta, se **SELECCIONA** **user\_id** de la tabla **transactions**, usando la letra **t** como alias y se agrupa igualmente por **user\_id**, usando **HAVING COUNT (t.user\_id)** y se realiza la comparación con mayor a (>) 30 para indicar que se debe filtrar según este criterio del número de transacciones.

## Exercici 2

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

```

--
17 • SELECT AVG(t.amount) AS mediana, cc.iban
18 FROM transactions t
19 JOIN companies c
20 ON t.business_id = c.company_id
21 JOIN credit_cards cc
22 ON cc.id = t.card_id
23 WHERE c.company_name = 'Donec Ltd'
24 GROUP BY cc.iban;

```

mediana	iban
203.715000	PT87806228135092429456346

Se escribe **SELECT AVG** de **transactions.amount** (donde se usa la letra **t** como alias) y se realiza un **JOIN** de **companies** (donde se usa la letra **c** como alias) donde **transaction.business id** es igual a **companies.company\_id**, se realiza otro **JOIN** de la tabla **credit\_cards** (donde se usa **cc** como alias) donde **credit\_cards.id** es igual a **transactions.card\_id** utilizando un **WHERE** para filtrar el nombre de la compañía **Donec Ltd**, por último se agrupa por **iban**.

## 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:

### Exercici 1

Quantes targetes estan actives?

## 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.