IT ACADEMY – DATA ANALYTICS

Alumna: Josselyn Maritza Jumpa Gordillo

SPRINT 4

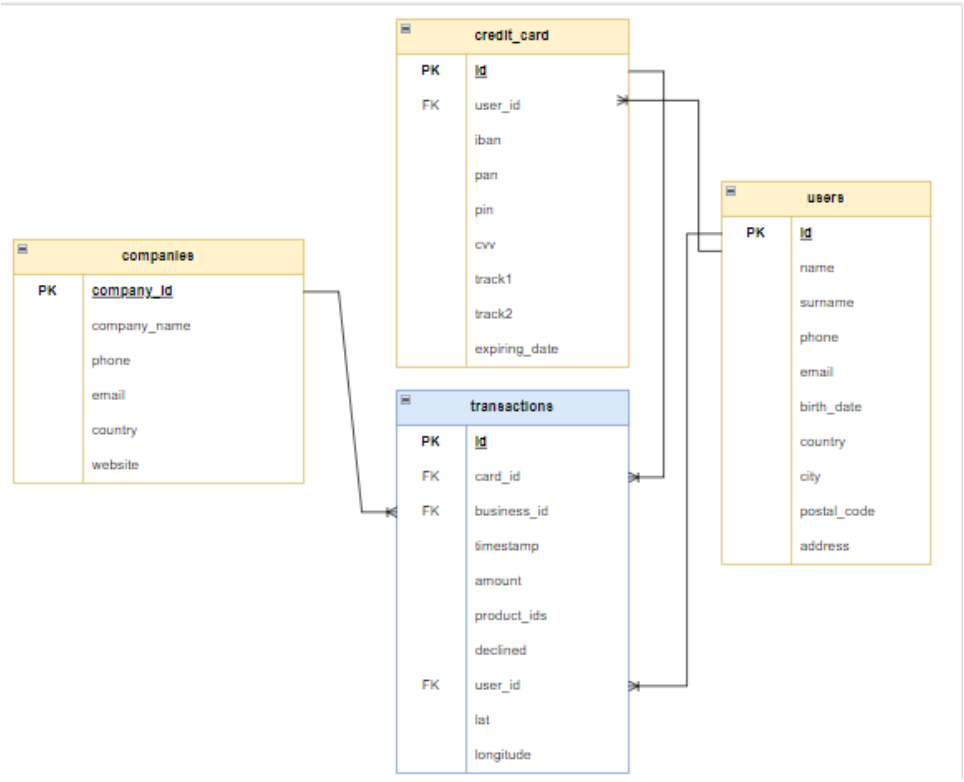
NIVEL 1

Descarrega 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:

Las 4 tablas que inicialmente componen la base de datos “business” son las siguientes:

- companies
- users (incluye usuarios de las regiones uk, ca, usa)
- credit_card
- transaction

El esquema de estrella que se ha diseñado es el siguiente:



A continuación, se comparten los comandos para la creación de la base de datos y tablas:

```sql
CREATE DATABASE IF NOT EXISTS business;

USE BUSINESS;

CREATE TABLE IF NOT EXISTS companies (
        company_id VARCHAR(20) PRIMARY KEY,
        company_name VARCHAR(255),
        phone VARCHAR(20),
        email VARCHAR(100),
        country VARCHAR(100),
        website VARCHAR(255)
    );

CREATE TABLE IF NOT EXISTS users(

        id INT PRIMARY KEY,
        name VARCHAR(100),
        surname VARCHAR(100),
        phone VARCHAR(150),
        email VARCHAR(150),
        birth_date date,
        country VARCHAR(150),
        city VARCHAR(150),
        postal_code VARCHAR(100),
        address VARCHAR(255)

    );

CREATE TABLE IF NOT EXISTS credit_card (
id varchar(20),
user_id int,
iban varchar(50),
pan varchar(50),
pin varchar(4),
cvv int,
track1 varchar(50),
track2 varchar(50),
expiring_date date,
primary key(id),
FOREIGN KEY(user_id) REFERENCES users(id)
);

CREATE TABLE IF NOT EXISTS transactions (
        id VARCHAR(255) PRIMARY KEY,
        card_id VARCHAR(20),
        business_id VARCHAR(20),
        timestamp TIMESTAMP,
        amount DECIMAL(10,2),
        declined boolean,
        product_ids VARCHAR(50),
        user_id INT,
        lat FLOAT,
        longitude FLOAT,
        FOREIGN KEY (card_id) REFERENCES credit_card(id),
        FOREIGN KEY (business_id) REFERENCES companies(company_id),
        FOREIGN KEY (user_id) REFERENCES users(id));
```

A continuación, se insertaron los datos a cada tabla. Esta es una muestra de cómo se añadieron los datos:

```
INSERT INTO companies (company_id, company_name, phone, email, country, website) VALUES (
'b-2222','Ac Fermentum Incorporated','06 85 56 52 33','donec.porttitor.tellus@yahoo.net','Germany','https://instagram.com/site');

INSERT INTO users (id, name, surname, phone, email, birth_date, country, city, postal_code, address) VALUES (
"1","Zeus","Gamble","1-282-581-0551","interdum.enim@protonmail.edu","1985-11-17","United States","Lowell","73544",
"348-7818 Sagittis St.");

INSERT INTO credit_card (id, user_id, iban, pan, pin, cvv, track1, track2, expiring_date) VALUES ('
'CcU-2938','275','TR301950312213576817638661','5424465566813633','3257','984','%B8383712448554646^WovsxejDpwiev^8604114?2?',
'%B7653863056044187=8007163336?3','2022-10-30');

INSERT INTO transactions (id, card_id, business_id, timestamp, amount, declined, product_ids, user_id, lat, longitude) VALUES ('
'108B1D1D-5B23-A76C-55EF-C568E49A05DD','CcU-2938','b-2222','2021-07-07 17:43:16','293.57','0',"59","275","83.7839152128",
,'-178.860353536');
```

Las 4 tablas se ven así:

companies:

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

users:

id	name	surname	phone	email	birth_date	country	city	postal_code	address
1	Zeus	Gamble	1-282-581-0551	interdum.enim@protonmail.edu	1985-11-17	United States	Lowell	73544	348-7818 Sagittis St.
2	Garrett	Mcconnell	(718) 257-2412	integer.vitae.nibh@protonmail.org	1992-08-23	United States	Des Moines	59464	903 Sit Ave

credit_card:

id	user_id	iban	pan	pin	cvv	track1	track2	expiring_date
CcU-2938	275	TR301950312213576817638661	5424465566813633	3257	984	%B8383712448554646^WovsxejDpwiev^8604...	%B7653863056044187=8007163336?3	2022-10-30
CcU-2945	274	DO26854763748537475216568689	5142423821948828	9080	887	%B4621311609958661^UftuyfsSeimxn^06106...	%B4149568437843501=5107140330?1	2023-08-24

transactions:

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	81.9185	-12.5276
0466A42E-47CF-8D24-FD01-C0B689713128	CcU-4219	b-2302	2021-07-26 07:29:18	49.53	0	47, 97, 43	170	-43.9695	-117.525

EJERCICIO 1

Enunciado:

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

Respuesta:

A continuación se muestra la consulta principal y subconsulta realizada empleando la tabla users y la tabla transactions:

```sql
SELECT users.*, num_transacciones
FROM users
-- Subquery para obtener ids y recuento de más de 30 transacciones
INNER JOIN(
SELECT user_id, COUNT(id) AS num_transacciones
FROM transactions
GROUP BY user_id
HAVING num_transacciones > 30) trans
-- unión de subquery con query principal
ON users.id = trans.user_id;
```

Estos son los usuarios con más de 30 transacciones:

id	name	surname	phone	email	birth_date	country	city	postal_code	address	num_transacciones
92	Lynn	Riddle	1-387-885-4057	vitae.aliquet@outlook.edu	1984-09-21	United States	Bozeman	61871	P.O. Box 712, 7907 Est St.	39
267	Ocean	Nelson	079-481-2745	aenean@yahoo.com	1991-12-26	Canada	Charlottetown	85X 3P4	Ap #732-8357 Pede, Rd.	52
272	Hedwig	Gilbert	064-204-8788	sem.eget@icloud.edu	1991-04-16	Canada	Tuktoyaktuk	Q4C 3G7	P.O. Box 496, 5145 Sapien ...	76
275	Kenyon	Hartman	082-871-7248	convallis.ante.lectus@yahoo.com	1982-08-03	Canada	Richmond	R8H 2K2	8564 Facilisi. St.	48

EJERCICIO 2

Enunciado:

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

Respuesta:

En la siguiente query se han empleado las tablas credit_card, transactions y companies.

```sql
-- obtención del IBAN, query principal
SELECT credit_card.iban, promedio_transacciones
FROM credit_card
INNER JOIN (-- obtención del promedio de transacciones de la empresa
SELECT card_id, AVG(amount) AS promedio_transacciones
FROM transactions
WHERE business_id IN (-- obtención del id de la empresa
SELECT company_id
FROM companies
WHERE company_name = 'Donec Ltd')
GROUP BY card_id) donec
ON credit_card.id = donec.card_id;
```

Este es el promedio de transacciones del IBAN correspondiente a la empresa Donec Ltd:

iban	promedio_transacciones
PT87806228135092429456346	203.715000

NIVEL 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:

En primer lugar, creamos la tabla card_status con dos campos, card_id y status_active:

```
CREATE TABLE IF NOT EXISTS card_status(
 card_id VARCHAR(20) PRIMARY KEY,
 status_active boolean,
FOREIGN KEY (card_id) REFERENCES credit_card(id));
```

Insertamos los datos en base a una query realizada en la tabla transactions.

Mediante Partition by, Row_number y Order by separamos la tabla transactions por tarjeta y asignamos un número de fila a cada transacción de cada tarjeta ordenándola de más a menos actual.

Subseleccionamos las tres primeras filas de cada tarjeta y si la suma de transacciones declinadas es 3 la consideramos inactiva y al revés activa.

```
INSERT INTO card_status (card_id, status_active)
-- creamos columna estado activo en base a la suma de la subquery
SELECT card_id, CASE WHEN SUM(declined)>=3 THEN 0 ELSE 1 END AS status_active
FROM
-- separamos la tabla por tarjeta, numeramos filas y ordenamos por fecha
(SELECT card_id, timestamp, declined,
ROW_NUMBER() OVER(PARTITION BY card_id ORDER BY timestamp DESC) AS Row_N
FROM
transactions) subquery_trans
WHERE Row_N <=3  -- seleccionamos sólo las tres primeras filas para cada tarjeta
GROUP BY card_id;
```

Así es como se ve la tabla card_status

card_id	status_active
CcU-2938	1
CcU-2945	1
CcU-2952	1
CcU-2959	1
CcU-2966	1
CcU-2973	1
CcU-2980	1
CcU-2987	1

EJERCICIO 1

Enunciado:

Quantes targetes estan actives?

Respuesta:

Hay un total de 275 tarjetas activas, todas las tarjetas empleadas en las transacciones.

```
SELECT COUNT(*) AS active_cards
FROM card_status
WHERE status_active = 1;
```

	active_cards
▶	275

NIVEL 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:

Primero, hemos creado la tabla products y hemos añadido los datos correspondientes:

```
CREATE TABLE IF NOT EXISTS products(
id VARCHAR (20) PRIMARY KEY,
product_name VARCHAR(255),
price DECIMAL(10,2),
colour VARCHAR(50),
weight int,
warehouse_id VARCHAR(50));
```

```
INSERT INTO products (id, product_name, price, colour, weight, warehouse_id) VALUES ('1','Direwolf Stannis','161.11','#7c7c7c',
'1','WH-4');
```

Una muestra de cómo se ve la tabla products:

id	product_name	price	colour	weight	warehouse_id
11	Karstark Dorne	49.70	#141414	27	WH--6
12	duel Direwolf	181.60	#a8a8a8	21	WH--7
13	palpatine chewbacca	139.59	#2b2b2b	1	WH--8
14	Direwolf	147.53	#c4c4c4	2	WH--9
15	Stannis warden	194.29	#dbdbdb	15	WH--10
16	the duel warden	180.91	#666666	3	WH--11

Luego, creamos una tabla auxiliar products_aux que llenaremos con información de la tabla transactions:

```
CREATE TABLE IF NOT EXISTS products_aux(
transaction_id VARCHAR(255),
p1 VARCHAR(20),
p2 VARCHAR(20),
p3 VARCHAR(20),
p4 VARCHAR(20));
```

Debido a que la tabla transactions tiene una columna donde se incluyen los id de productos separados por coma, en esta tabla auxiliar los separamos en distintas columnas:

```
INSERT INTO products_aux(transaction_id, p1, p2,p3,p4)

SELECT id as transaction_id,
trim(substring(replace(product_ids, ',', ' '),1,2)) as p1,
trim(substring(replace(product_ids, ',', ' '),3,4)) AS p2,
trim(substring(replace(product_ids, ',', ' '),7,4)) AS p3,
trim(substring(replace(product_ids, ',', ' '),11,4)) AS p4
FROM transactions;
```

Así se ve la tabla auxiliar products_aux

id	p1	p2	p3	p4
02C6201E-D90A-1859-B4EE-88D2986D3B02	71	1	19	
0466A42E-47CF-8D24-FD01-C0B689713128	47	97	43	
063FBA79-99EC-66FB-29F7-25726D1764A5	47	67	31	5
0668296C-CDB9-A883-76BC-2E4C44F8C8AE	89	83	79	
06CD9AA5-9B42-D684-DDDD-A5E394FEBA99	43	31		
07A46D48-31A3-7E87-65B9-0DA902AD109F	47	23		
09DE92CE-6F27-2BB7-13B5-9385B2B3B8E2	67	7		

En esta ocasión, crearemos la tabla transaction_product que tendrá integridad referencial con las tablas transactions y products:

```
CREATE TABLE IF NOT EXISTS transaction_product(
transaction_id VARCHAR(255),
product_id VARCHAR(20),
FOREIGN KEY (transaction_id) REFERENCES transactions(id),
FOREIGN KEY (product_id) REFERENCES products(id));
```

Utilizaremos la tabla auxiliar creada anteriormente para rellenar la tabla transaction_product mediante UNION ALL de cada columna con producto

```
INSERT INTO transaction_product(transaction_id, product_id)
SELECT transaction_id, p1 AS product_id
FROM products_aux
UNION ALL
SELECT transaction_id, p2 AS product_id
FROM products_aux
WHERE length(p2) > 0
UNION ALL
SELECT transaction_id, p3 AS product_id
FROM products_aux
WHERE length(p3) > 0
UNION ALL
SELECT transaction_id, p4 AS product_id
FROM products_aux
WHERE length(p4) > 0;
```

La tabla transaction_product se ve así:

transaction_id	product_id
02C6201E-D90A-1859-B4EE-88D2986D3B02	71
02C6201E-D90A-1859-B4EE-88D2986D3B02	1
02C6201E-D90A-1859-B4EE-88D2986D3B02	19
0466A42E-47CF-8D24-FD01-C0B689713128	47
0466A42E-47CF-8D24-FD01-C0B689713128	43
0466A42E-47CF-8D24-FD01-C0B689713128	97

Ya podemos eliminar la tabla auxiliar creada:

```
DROP table products_aux;
```

EJERCICIO 1

Enunciado:

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

Respuesta:

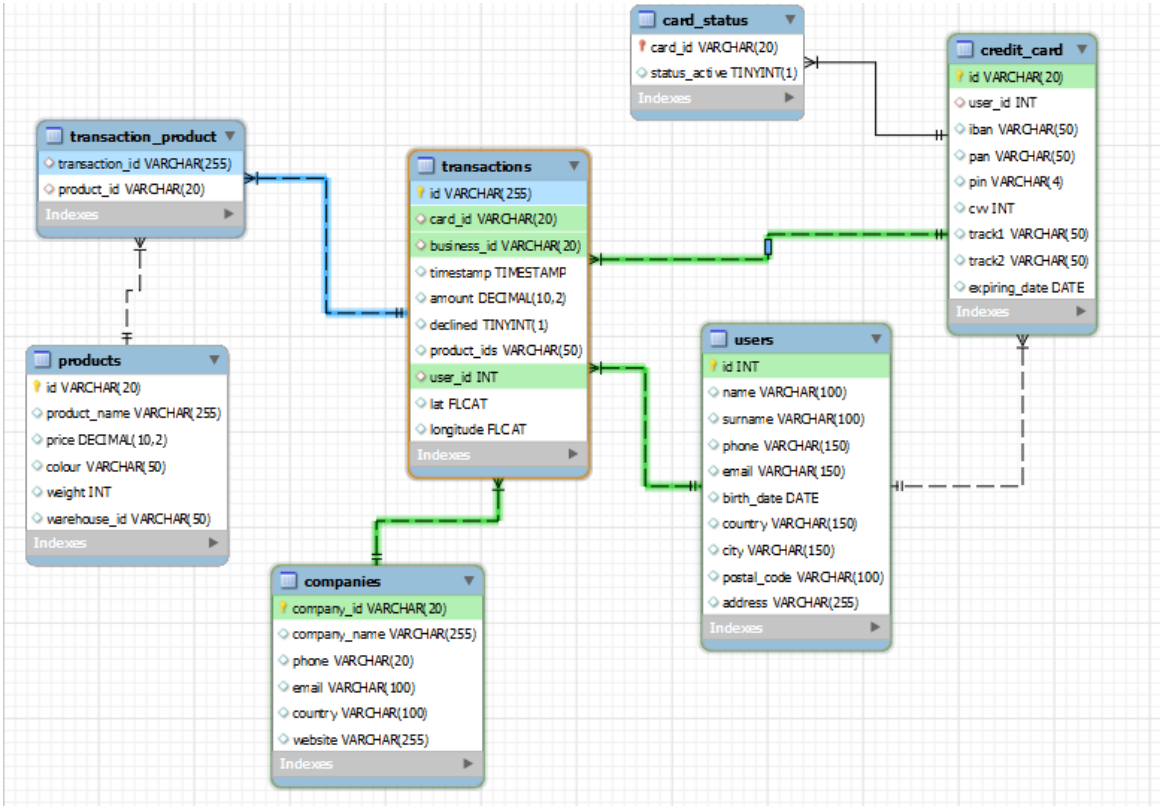
Contamos el número de veces que se repite cada artículo y obtenemos el nombre del producto:

```
SELECT product_name, COUNT(product_id) as num_of_purchases
FROM transaction_product
INNER JOIN products
ON transaction_product.product_id = products.id
GROUP BY product_id
ORDER BY num_of_purchases DESC;
```

A continuación, se muestra las primeras filas de resultados:

product_name	num_of_purchases
riverlands north	68
Winterfell	68
Direwolf riverlands the	66
Tarly Stark	65
duel	65
Tully	62
Direwolf Stannis	61
skywalker ewok sith	61
jinn Winterfell	61
palpatine chewbacca	60

Finalmente, el esquema de nuestra base de datos ha quedado así:


card_status
card_id VARCHAR(20)
status_active TINYINT(1)
Indexes
credit_card
id VARCHAR(20)
user_id INT
iban VARCHAR(50)
pan VARCHAR(50)
pin VARCHAR(4)
cvv INT
track1 VARCHAR(50)
track2 VARCHAR(50)
expiring_date DATE
Indexes
transaction_product
transaction_id VARCHAR(255)
product_id VARCHAR(20)
Indexes
transactions
id VARCHAR(255)
card_id VARCHAR(20)
business_id VARCHAR(20)
timestamp TIMESTAMP
amount DECIMAL(10,2)
declined TINYINT(1)
product_ids VARCHAR(50)
user_id INT
lat FLOAT
longitude FLOAT
Indexes
users
id INT
name VARCHAR(100)
surname VARCHAR(100)
phone VARCHAR(150)
email VARCHAR(150)
birth_date DATE
country VARCHAR(150)
city VARCHAR(150)
postal_code VARCHAR(100)
address VARCHAR(255)
Indexes
products
id VARCHAR(20)
product_name VARCHAR(255)
price DECIMAL(10,2)
colour VARCHAR(50)
weight INT
warehouse_id VARCHAR(50)
Indexes
companies
company_id VARCHAR(20)
company_name VARCHAR(255)
phone VARCHAR(20)
email VARCHAR(100)
country VARCHAR(100)
website VARCHAR(255)
Indexes