Sprint 4: Creación de bases de datos en SQL

Curso de especialización Data Analytics - IT Academy - Barcelona Activa

Nicolás Aros Marzá

13/01/2026

Nivel 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:.

La primera acción ejecutada en el script es la creación del entorno de trabajo mediante la instrucción `CREATE DATABASE transacciones_usuarios`. Posteriormente, se define la estructura de cada tabla y variable a través de la instrucción `CREATE TABLE` (`users`, `companies`, `credit_cards`, `products` y `transactions`).

Creación de la bbdd y las tablas users, companies y credit_cards.

```sql
CREATE DATABASE transacciones_usuarios;
USE transacciones_usuarios;

CREATE TABLE users (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR (20),
    surname VARCHAR (50),
    phone VARCHAR (50),
    email VARCHAR (100),
    birth_date VARCHAR (50),
    country VARCHAR (20),
    city VARCHAR (50),
    postal_code VARCHAR (20),
    address VARCHAR (100)
);

CREATE TABLE companies (
    company_id VARCHAR (20) NOT NULL PRIMARY KEY,
    company_name VARCHAR (50),
    phone VARCHAR (50),
    email VARCHAR (50),
    country VARCHAR (20),
    website VARCHAR (50)
);

CREATE TABLE credit_cards (
    id VARCHAR (50) NOT NULL PRIMARY KEY,
    user_id SMALLINT,
    iban VARCHAR (50),
    pan VARCHAR (50),
    pin BIGINT,
    cvv SMALLINT,
    track1 VARCHAR (100),
    track2 VARCHAR (100),
    expiring_date VARCHAR (10)
);
```

En la creación de la tabla `transactions` se realiza un paso adicional al definir la relación con las otras tablas mediante el uso de `FOREIGN KEY`. Estas claves establecen una relación de uno a muchos (1:N), vinculando cada transacción con un usuario, una empresa y una tarjeta específica.

Creación de la tabla products y transactions.

```sql
CREATE TABLE products (
    id VARCHAR (100) NOT NULL PRIMARY KEY,
    product_name VARCHAR (50),
    price VARCHAR(20),
    colour VARCHAR (20),
    weight DECIMAL (10,1),
    warehouse_id VARCHAR (10)
    );

CREATE TABLE transactions (
    id VARCHAR (50) NOT NULL PRIMARY KEY,
    card_id VARCHAR (10),
    business_id VARCHAR (20),
    timestamp TIMESTAMP,
    amount DECIMAL (10,2),
    declined BOOLEAN,
    product_id VARCHAR (100),
    user_id INT,
    lat FLOAT,
    longitude FLOAT,
        FOREIGN KEY (user_id) REFERENCES users (id),
        FOREIGN KEY (business_id) REFERENCES companies (company_id),
        FOREIGN KEY (card_id) REFERENCES credit_cards (id)
    );

```

Para continuar con el proceso, se habilita la importación de archivos locales con el comando `SET GLOBAL local_infile = 1`. Luego, se utiliza la instrucción `LOAD DATA LOCAL INFILE` para poblar las tablas desde archivos CSV.

Importación de datos CSV a las tablas.

```
66 •  SET GLOBAL local_infile = 1; -- Permitir la importación de archivos locales
67 •  SHOW GLOBAL VARIABLES LIKE 'local_infile'; -- Confirmar si el cambio se realizó correctamente
68
69    -- TABLA USUARIOS (american_users)
70 •  LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/american_users.csv'
71    INTO TABLE users
72    FIELDS TERMINATED BY ','
73    OPTIONALLY ENCLOSED BY '"'
74    LINES TERMINATED BY '\n'
75    IGNORE 1 LINES;
76
77    -- TABLA USUARIOS (european_users)
78 •  LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/european_users.csv'
79    INTO TABLE users
80    FIELDS TERMINATED BY ','
81    OPTIONALLY ENCLOSED BY '"'
82    LINES TERMINATED BY '\n'
83    IGNORE 1 LINES;
84
85    -- TABLA COMPANIES
86 •  LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/companies.csv'
87    INTO TABLE companies
88    FIELDS TERMINATED BY ','
89    LINES TERMINATED BY '\n'
90    IGNORE 1 LINES;
91
92    -- TABLA CREDIT_CARD
93 •  LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/credit_cards.csv'
94    INTO TABLE credit_cards
95    FIELDS TERMINATED BY ','
96    LINES TERMINATED BY '\n'
97    IGNORE 1 LINES;
```

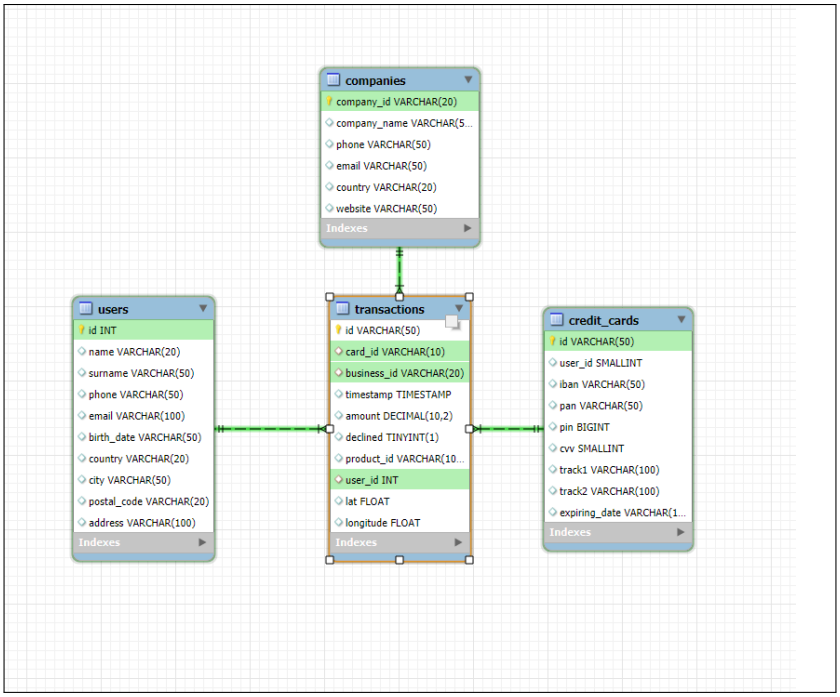
Aunque se usan las mismas instrucciones de carga en todas las tablas, en `credit_cards` y `products` se aplicaron ajustes adicionales. Estos cambios permiten normalizar los datos, convirtiendo las fechas a formato `DATE` y transformando los precios en valores numéricos. Para esto, se desactivó temporalmente el modo seguro con `SET SQL_SAFE_UPDATES = 0`.

Importación de datos CSV y actualización las tablas.

```sql
SET SQL_SAFE_UPDATES = 0;

UPDATE credit_cards
SET expiring_date = STR_TO_DATE(expiring_date, '%m/%d/%y');
ALTER TABLE credit_cards MODIFY expiring_date DATE;

-- TABLA PRODUCTS
LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/products.csv'
INTO TABLE products
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

UPDATE products
SET price = REPLACE(price, '$', '');
ALTER TABLE products MODIFY price DECIMAL(10,2);

-- TABLA TRANSACTIONS
LOAD DATA LOCAL INFILE 'G:/Mi unidad/_dataanalytics_2025/S4_SQL/transactions.csv'
INTO TABLE transactions
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

SET SQL_SAFE_UPDATES = 1; -- Retornar al modo seguro
```

Tras la carga de los datos, hasta este momento, el esquema resultante tiene esta forma de estrella.

Visualización de la base de datos transacciones_usuarios.



Ejercicio 1

Realitza una subconsulta que mostri tots els usuaris amb més de 80 transaccions utilitzant almenys 2 taules.

Consulta para identificar usuarios con más de 80 transacciones.

```
127 •  SELECT t.user_id, COUNT(DISTINCT(t.id)) AS cuenta_transacciones
128    FROM transactions t
129    WHERE t.user_id IN (
130        SELECT id
131        FROM users u
132        )
133    GROUP BY t.user_id
134    HAVING cuenta_transacciones > 80
135    ORDER BY t.user_id DESC;
```

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

user_id	cuenta_transacciones
454	81
318	91
289	94
185	110

La consulta principal agrupa las transacciones por usuario y se utiliza la cláusula `HAVING` para filtrar los resultados, mostrando únicamente a aquellos que superan las 80 transacciones. El resultado final se presenta ordenado por ID de forma descendente.

Ejercicio 2

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

Consulta para conocer el promedio de cada tarjeta Donec Ltd.

```
139 •  SELECT cc.iban, ROUND(AVG(t.amount),2) AS promedio_monto, c.company_name
140    FROM transactions t
141    INNER JOIN credit_cards cc ON t.card_id = cc.id
142    INNER JOIN companies c ON t.business_id = c.company_id
143    WHERE c.company_name = 'Donec Ltd'
144    GROUP BY cc.iban
145    ORDER BY ROUND(promedio_monto,2) DESC;
```

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

iban	promedio_monto	company_name
XX3830178139196201993663 52	680.69	Donec Ltd
XX6377063573975703949739 13	680.01	Donec Ltd
XX9713939714652922023122 59	645.46	Donec Ltd
XX1718471169288923759693 07	628.89	Donec Ltd
XX2254246388185424062235 75	608.68	Donec Ltd
XX7488907290571957117660 71	607.29	Donec Ltd
TN9614563570667381893122	605.41	Donec Ltd
XX4819080340373642425911 85	605.36	Donec Ltd
XX1946755197392563357535 08	597.19	Donec Ltd
XX2159627660619671954934 37	594.26	Donec Ltd
XX4493223208268907210014 43	591.61	Donec Ltd
XX5351854927357042294742 37	570.09	Donec Ltd
CH9552373968796160224	566.38	Donec Ltd
XX3476053771256378803031 31	561.80	Donec Ltd
XX6884714466979219128603 04	543.42	Donec Ltd
XX6055339645824587041059 56	542.00	Donec Ltd
PL7624928356685267634340 4	541.56	Donec Ltd

Es este código primero se vinculan las tablas `transactions`, `credit_cards` y `companies` mediante `INNER JOIN`. Luego, se filtran las observaciones que corresponden a la empresa "Donec Ltd". El promedio se calcula con la función `AVG` y se agrega `ROUND` para ajustar el resultado a dos decimales. Finalmente, agrupa los datos por `IBAN`.

Nivel 2

Ejercicio 1

Crea una nova taula que reflecteixi l'estat de les targetes de crèdit basat en si les tres últimes transaccions han estat declinades aleshores és inactiu, si almenys una no és rebutjada aleshores és actiu. Partint d'aquesta taula respon:

Quantes targetes estan actives?

Primero, el script crea la tabla `estado_tarjeta`. Para ordenar las transacciones según las más recientes hasta las mas antiguas, se utiliza la función `ROW_NUMBER()`. Esto permite identificar las tres últimas transacciones de cada tarjeta. Mediante la función `SUM` se crea la variable `cuenta_rechazos`, que cuenta el número de esas transacciones que han sido rechazadas.

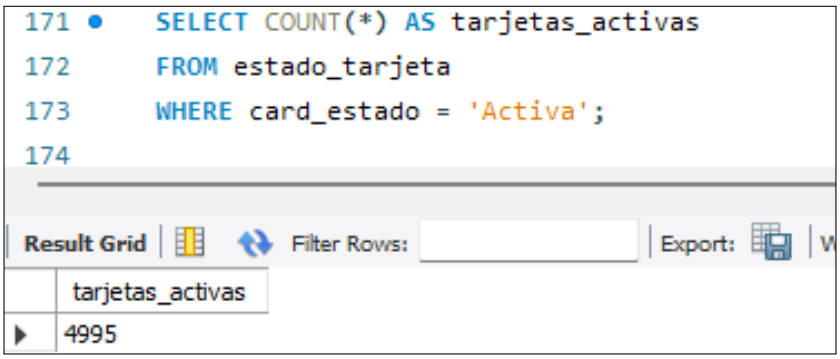
Con esta variable se identifican las tarjetas según si están activas o no. Para esto, se usa la instruccion `CASE`. El criterio es que si la variable cuenta_rechazos es igual a tres, la tarjeta se marca como "Inactiva". Los casos en que no se cumpla esta condición, se etiquetan como "Activa".

Creación de la tabla estado_tarjeta.

```sql
-- Nivel 2: ejercicio 1

CREATE TABLE estado_tarjeta AS
SELECT
card_id,
    CASE
        WHEN cuenta_rechazos = 3 THEN 'Inactiva'
        ELSE 'Activa'
    END AS card_estado
FROM (
    SELECT
        card_id,
        SUM(CASE WHEN declined = '1' THEN 1 ELSE 0 END) AS cuenta_rechazos
    FROM (
        SELECT
            card_id,
            declined,
            ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY timestamp DESC) AS num_fila
        FROM transactions
    ) AS t_orden
    WHERE num_fila <= 3
    GROUP BY card_id
) AS t_resumen;
```

Mediante `SELECT COUNT` sobre esta nueva tabla para obtener el número total de tarjetas que cumplen con el estado activo.

Consulta para mostrar el número de tarjetas activas.



Nivel 3

Ejercicio 1

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:

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

El primer paso es crear la tabla `products_nueva`. Debido a que una transacción puede contener varios artículos y están alojados en una sola columna en formato texto, se utiliza la función `REPLACE` para convertirla en una lista. Luego, mediante `JSON_TABLE`, se desglosan estos elementos en filas individuales, permitiendo que cada producto se vincule correctamente a su transacción.

Posteriormente, se ejecuta una consulta que une la tabla intermedia con products y transactions mediante INNER JOIN. El código aplica un filtro para contabilizar únicamente las operaciones exitosas (declined = 0), agrupando los resultados por el nombre del artículo. Finalmente, utiliza la función COUNT para obtener el número total de ventas de cada producto, presentando la lista ordenada de mayor a menor.

Creación de la tabla products_nueva.

```
175    -- Nivell 3 - Exercici 1:
176
177    CREATE TABLE products_nueva (
178      transaction_id VARCHAR (50) NOT NULL,
179      products_nueva_id VARCHAR (100) NOT NULL,
180      PRIMARY KEY (transaction_id, products_nueva_id),
181      FOREIGN KEY (transaction_id) REFERENCES transactions(id),
182      FOREIGN KEY (products_nueva_id) REFERENCES products(id)
183      );
184
185    INSERT INTO products_nueva (transaction_id, products_nueva_id)
186    SELECT
187      t.id AS transaction_id,
188      TRIM(aux.product_id) AS products_nueva_id
189    FROM transactions t
190    JOIN JSON_TABLE(
191      CONCAT('["', REPLACE(t.product_id, ',', '","'), '"]'),
192      '$[*]' COLUMNS (product_id VARCHAR(100) PATH '$')
193    ) AS aux
194    JOIN products p ON p.id = TRIM(aux.product_id);
```

Posteriormente, se ejecuta para obtener el número total de ventas de cada producto.

Consulta para mostrar el número veces que se ha vendido cada producto.

```sql
SELECT
    pn.products_nueva_id,
    p.product_name,
    COUNT(*) AS numero_ventas
FROM products_nueva pn
INNER JOIN products p ON pn.products_nueva_id = p.id
INNER JOIN transactions t ON pn.transaction_id = t.id
WHERE t.declined = 0
GROUP BY pn.products_nueva_id, p.product_name
ORDER BY numero_ventas DESC;
```

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

products_nueva_id	product_name	numero_ventas
52	riverlands the duel	2642
29	Tully maester Tarly	2627
21	duel Direwolf	2603
16	the duel warden	2602
33	duel warden	2593
87	sith Jade	2591
66	mustafar jinn	2590
48	rock Renly in	2589
68	Stark Karstark	2587
23	riverlands north	2586
88	Stannis warden so...	2582
73	Dorne bastard	2579

Result 6