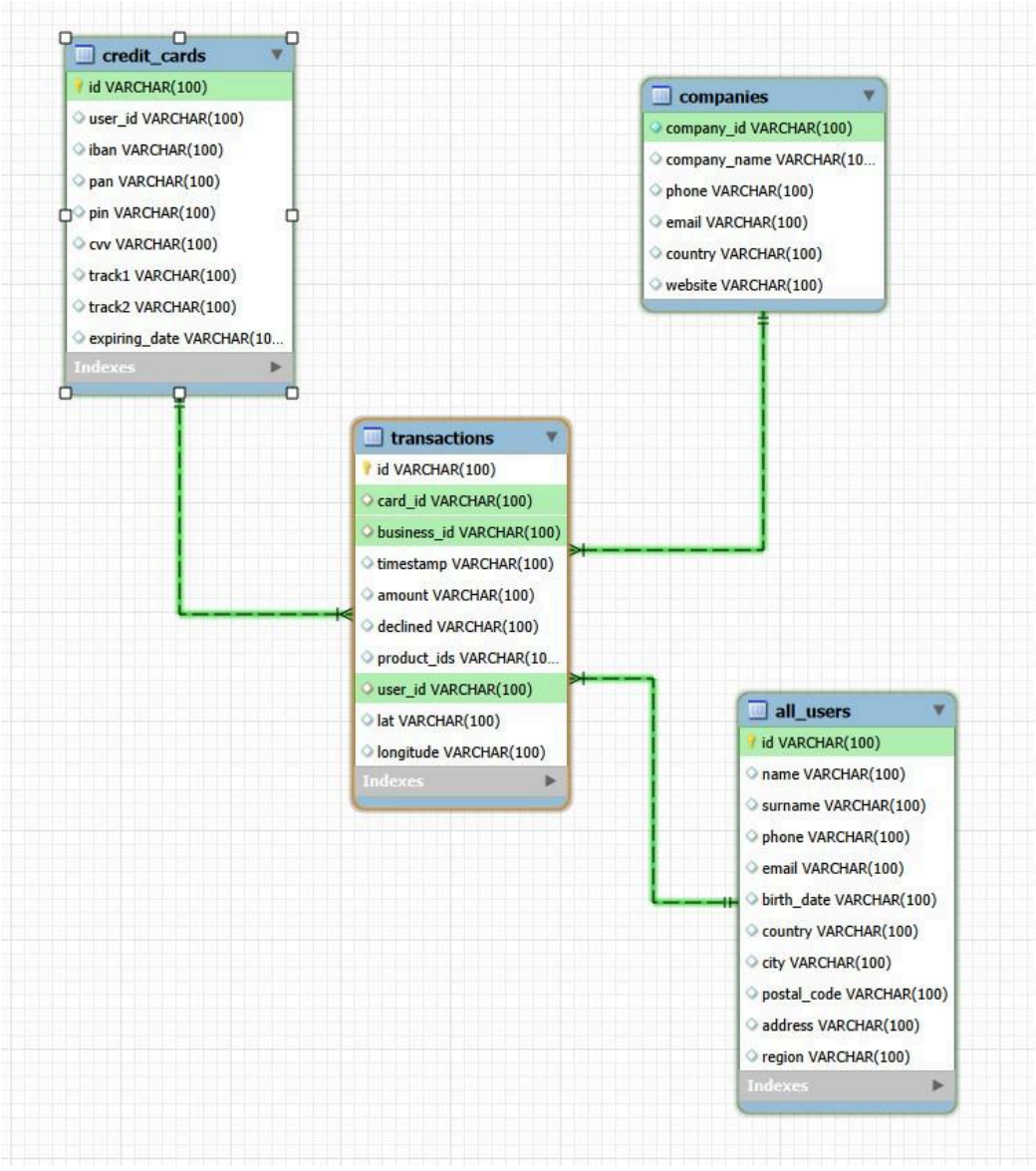
Data Analytics - SQL
Sprint 4 - Mariana P.

Nivel 1

Descarga los archivos CSV, estudiales y diseña una base de datos con un **esquema de estrella que contenga, al menos 4 tablas** de las que puedas realizar las siguientes consultas:



COMENTARIOS:
Esta es una Tabla de hecho tipo **Transacción**, porque registra una fila por transacción.
El Tipo de entidad: Evento transaccional
Rol: Registra hechos medibles conectados a dimensiones. En este ejemplo: usuario, empresa y tarjeta de crédito.

Ejercicio 1

Realiza una subconsulta que muestre a todos los usuarios con más de 80 transacciones utilizando al menos 2 tablas.

COMENTARIOS:

- Utilizo las tablas ‘transactions’ y la tablas ‘all_users’ que es la junción de las tablas american_users y european_users.
- el mismo ID de all_users puede tener muchas transacciones.
- COUNT(t.id) es una función agregada que se calcula por grupo
- HAVING filtra los grupos con más de 80 transacciones

- Aquí puedo ver los 4 usuarios que tienen más de 80 transacciones:

```
272     -- Nivel 1
273     -- Exercici 1
274     -- Realiza una subconsulta que muestre a todos los usuarios con más de 80 transacciones utilizando al menos 2 tablas.
275
276 •   USE sprint_4;
277
278     -- Total de usuarios con más de 80 transaciones:
279 •   SELECT *
280     FROM all_users
281     WHERE id IN (
282         SELECT user_id
283         FROM transactions
284         GROUP BY user_id
285         HAVING COUNT(*) > 80
286     );
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	surname	phone	email	birth_date	country	city	postal_code	address	region
185	Molly	Gilliam	0800 120 8023	donec@outlook.couk	Dec 21, 1993	United Kingdom	London	EC1A 1BB	P.O. Box 202, 5638 Mi Rd.	european
289	Dxwgi	Hwcru	+98-309-8797	dxwgi.hwcru@example.com	Aug 20, 1976	Germany	Stuttgart	70173	82 Hwcru Street	european
318	Bnyr	Astuw	+33-120-9644	bnyr.astuw@example.com	May 3, 1974	Italy	Genoa	16100	53 Astuw Street	european
454	Sfzzoh	Xgvfridxs	+58-495-3945	sfzzoh.xgvfridxs@example.com	Aug 28, 1962	Poland	Gdansk	80-001	52 Xgvfridxs Street	european
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Aquí puedo ver cuantas transacciones cada uno de los 4 realizó:

```
287
288     -- Numero de transaciones totales por cada uno de los 4 usuarios:
289
290 •   SELECT au.id, au.name, au.surname,
291         COUNT(t.id) AS total_transactions
292     FROM all_users au
293     JOIN transactions t ON au.id = t.user_id
294     GROUP BY au.id, au.name, au.surname
295     HAVING COUNT(t.id) > 80;
```

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

id	name	surname	total_transactions
185	Molly	Gilliam	110
289	Dxwgi	Hwcru	94
318	Bnyr	Astuw	91
454	Sfzzoh	Xgvfridxs	81

Ejercicio 2

Muestra la media de amount por IBAN de las tarjetas de crédito en la compañía Donec Ltd., utiliza por lo menos 2 tablas.

-- Primero confirmar el nombre de la compañía
-- NOTA: es Donec Ltd sin el punto.

```
313 •   SELECT company_name
314     FROM companies
315     WHERE company_name = 'Donec Ltd';
```

Result Grid | Filter Rows: | Export:

company_name
Donec Ltd

Opción 1

```
317
318     -- Aquí quiero ordenar desde la mayor media por amount
319
320 •   SELECT c.company_name, cc.iban, AVG(t.amount)
321     FROM transactions t
322     JOIN credit_cards cc ON t.card_id = cc.id
323      JOIN companies c ON t.business_id = c.company_id
324      WHERE c.company_name = 'Donec Ltd'
325     GROUP BY cc.iban, company_name
326     ORDER BY AVG(t.amount) DESC;
```

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

company_name	iban	AVG(t.amount)
Donec Ltd	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

Opción 2

```
327
328     -- Y aquí, ordenar desde la mayor media por el iban
329 •   SELECT c.company_name, cc.iban, AVG(t.amount)
330     FROM transactions t
331     JOIN credit_cards cc ON t.card_id = cc.id
332      JOIN companies c ON t.business_id = c.company_id
333      WHERE c.company_name = 'Donec Ltd'
334     GROUP BY cc.iban, company_name
335     ORDER BY cc.iban;
336
```

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

company_name	iban	AVG(t.amount)
Donec Ltd	AD2777204763277722050982	214.48
Donec Ltd	AE491827142302887266369	262.12
Donec Ltd	AL65255766650838829611407660	169.96
Donec Ltd	AT461138859877579187	303.385
Donec Ltd	AT658218806585843788	57.25
Donec Ltd	AZ03248993733987407386489271	382.86
Donec Ltd	AZ39336002925842865843941994	419.55
Donec Ltd	AZ58645632361051026278861261	207.84

Nivel 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basado en si las tres últimas transacciones han sido declinadas entonces es inactivo, si al menos una no es rechazada entonces es activo . Partiendo un esta tabla responde:

-- declinadas = 1 = INACTIVO
-- declinadas = 0 = ACTIVO

PASO 1: Crear la nueva tabla con los datos que me van a ayudar a identificar el estado de las tarjetas

```
348 • CREATE TABLE IF NOT EXISTS card_status (
349       card_id VARCHAR(50) PRIMARY KEY,
350       status VARCHAR(50),
351       declined VARCHAR (2)
352   );
353
```

PASO 2: Insertar datos de la columna id de la tabla 'credit_card' en la columna card_id de la tabla 'card_status':

```
355
356 • INSERT INTO card_status (card_id)
357   SELECT id FROM credit_cards;
358
```

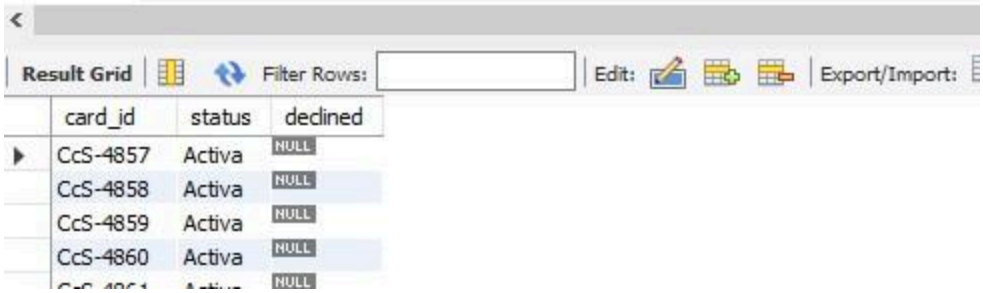
PASO 3: Actualizar la tabla card_status cs con datos de las columnas de transaction t con subconsultas para definir y agrupar los datos 'declined' de las 3 últimas fechas:

COMENTARIOS:

- Fue necesario crear una tabla temporal y una columna temporal para poder relacionar la base de datos ya existente con la que se quería identificar.
- CASE : usado por que cría una condición
- WHEN: cuando la siguiente ecuación se realice:
  SUM(t.declined) : suma los valores de la columna ‘declined’
- = 3 THEN 'Inactiva': si las sumas totales de las 3 ultimas fila (las 3 últimas fechas) son = a 3 entonces significa que es INACTIVA.
- ELSE '**Activa**': todas las demás son ACTIVAS
- END AS **new_status:** que termine con el alias **‘new_status’**
- FROM transactions t: que sale de la tabla transactions

- SELECT COUNT**(*)**: Cuenta cuántas transacciones posteriores existen para la misma tarjeta.
- FROM transactions **tp2**: instancia temporal de la tabla transactions que puse el alias como ‘tp2’
- WHERE **tp2**.card_id = t.card_id: los datos de tp2 y transactions son los mismos para card_id.
- AND **tp2**.timestamp > t.timestamp) **< 3**:busca transacciones que ocurrieron después de la actual (t) para que yo pueda saber si t está entre las últimas 3 transacciones.
- GROUP BY t.card_id (): Agrupa las transacciones por tarjeta para poder ejecutar SUM(t.declined) por tarjeta.
- AS last_3_dates: tabla virtual para la subconsulta, donde contiene: card_id y new_status para cada tarjeta.
- ON cs.card_id = last_3_dates.card_id: Une la tabla card_status con la tabla virtual last_3_dates.
- SET cs.status = last_3_dates.new_status: Actualiza el campo status en card_status con el nuevo estado calculado.
- WHERE **cs.card_id != '':** el commando ‘diferente de nada’ ayuda a evitar tarjetas con ID vacío.

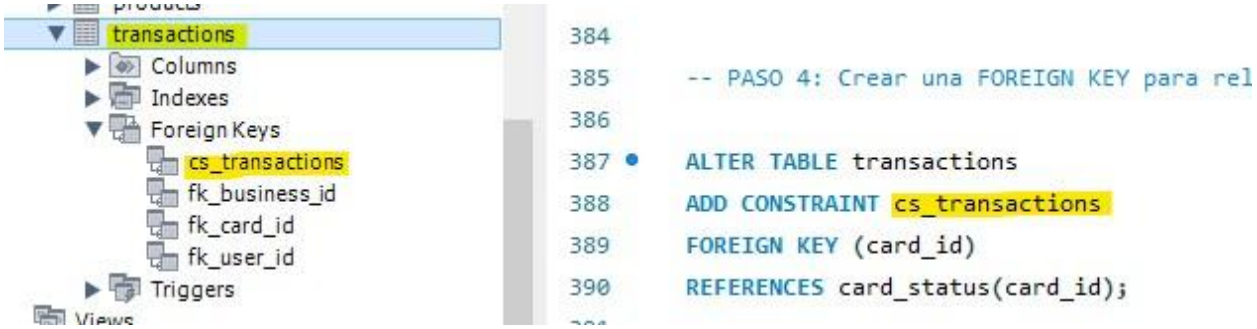
```sql
UPDATE card_status cs
JOIN (
    SELECT t.card_id,
           CASE
               WHEN SUM(t.declined) = 3 THEN 'Inactiva'
               ELSE 'Activa'
           END AS new_status
    FROM transactions t
    WHERE (
        SELECT COUNT(*)
        FROM transactions tp2
        WHERE tp2.card_id = t.card_id
          AND tp2.timestamp > t.timestamp
    ) < 3
    GROUP BY t.card_id
    ) AS last_3_dates
ON cs.card_id = last_3_dates.card_id
SET cs.status = last_3_dates.new_status
WHERE cs.card_id != '';

SELECT * FROM card_status;

-- COMENTARIOS:
```



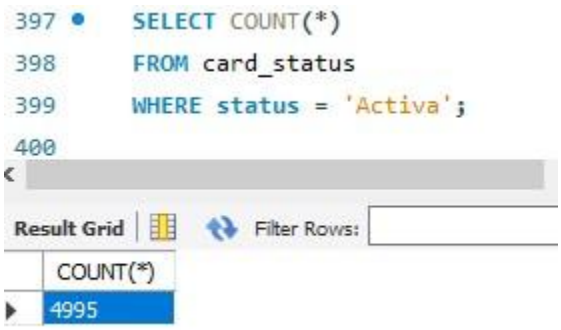
Result Grid | Filter Rows: | Edit: | Export/Import:

card_id	status	declined
CcS-4857	Activa	NULL
CcS-4858	Activa	NULL
CcS-4859	Activa	NULL
CcS-4860	Activa	NULL

PASO 4: Crear una FOREIGN KEY para relacionar la tabla 'transactions' con la tabla 'card_status'



PASO 5 - Ejercicio 1
¿Cuántas tarjetas están activas?



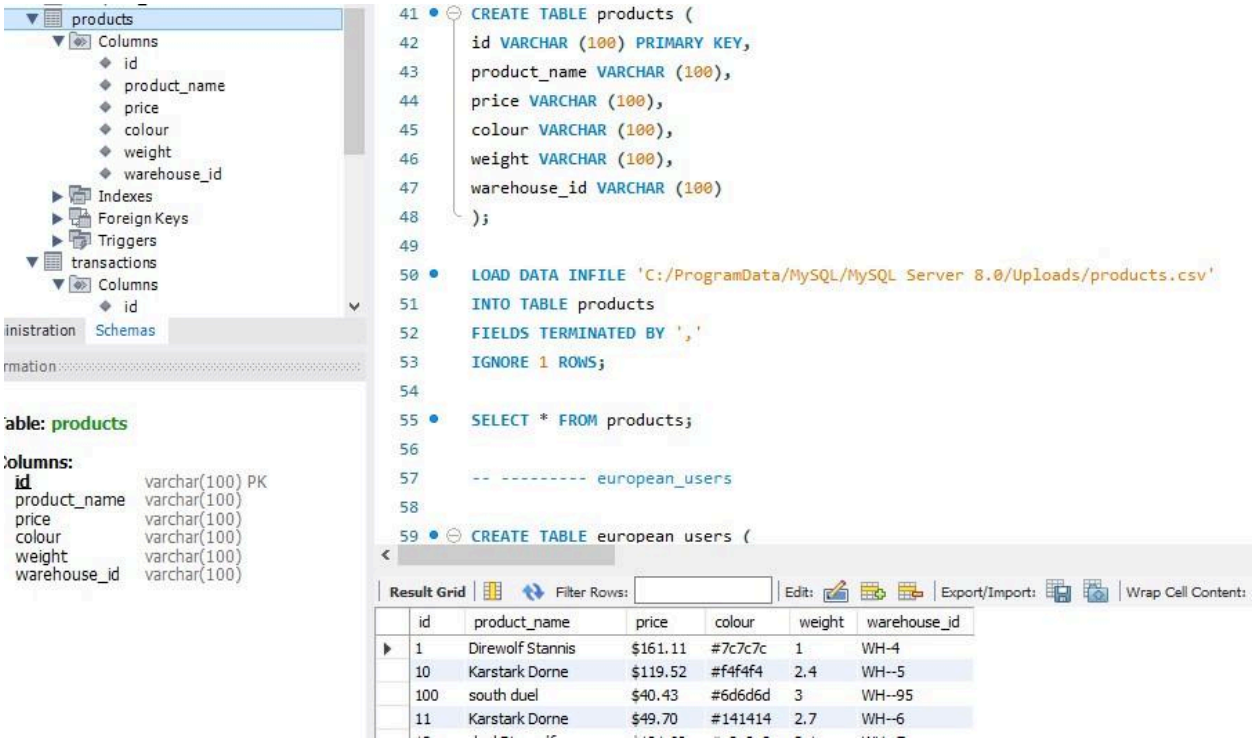
COMENTARIO:

- Selecciono COUNT(*) para sumar todas las activas = 4995

Nivel 3

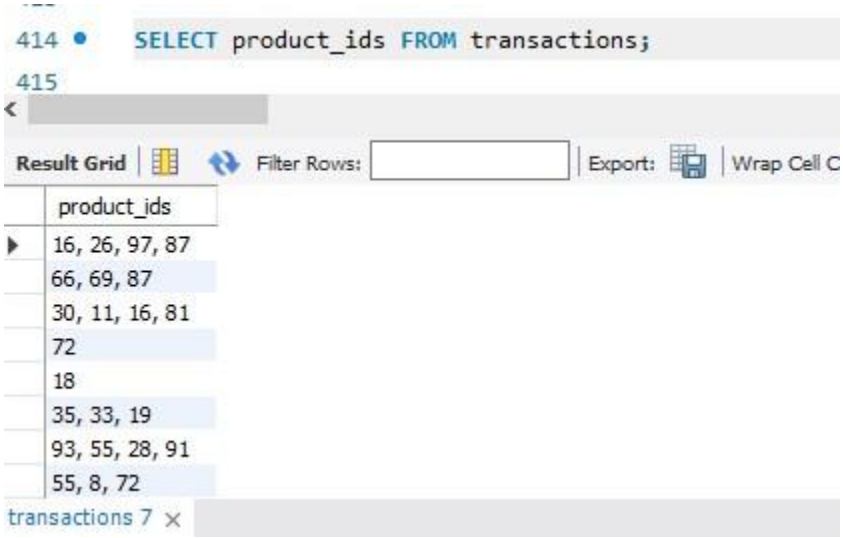
Crea una tabla con la que podamos unir los datos del nuevo archivo products.csv con la base de datos creada, teniendo en cuenta que desde **transaction** tienes **product_ids**. Genera la siguiente consulta:

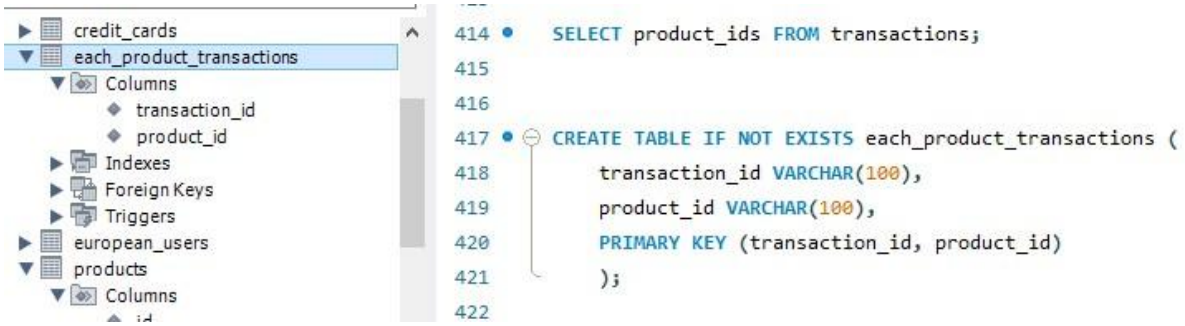
PASO 1: Crear la nueva tabla ‘products’ e insertar la base de datos csv. correspondiente:



PASO 2: Crear una ‘Tabla de Puente’ por ser el caso de una relación entre tablas de muchos a muchos:

- 1 transacción tiene muchos productos.
- 1 producto tiene muchas transacciones.
- Por haber varios ids de productos dentro de una misma fila, es necesario crear una Primary Key compuesta.





PASO 3: Para separar cada producto de cada fila en diferentes columnas, se necesita utilizar JSON*

```
426  INSERT INTO each_product_transactions (transaction_id, product_id)
427  SELECT t.id, CAST(j.value AS UNSIGNED) AS product_id
428  FROM transactions t
429  JOIN JSON_TABLE(
430      CONCAT(
431          '["',
432          REPLACE(t.product_ids, ',', '","'),
433          '"]'
434      ),
435      '$[*]' COLUMNS (value VARCHAR(50) PATH '$')
436  ) AS j;
437
438
439  SELECT * FROM each_product_transactions;
440
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Ce

transaction_id	product_id
00043A49-2949-494B-A5DD-A5BAE3BB19DD	16
00043A49-2949-494B-A5DD-A5BAE3BB19DD	26
00043A49-2949-494B-A5DD-A5BAE3BB19DD	87
00043A49-2949-494B-A5DD-A5BAE3BB19DD	97
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	66
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	69
000447FE-B650-4DCF-85DE-C7ED0EE1CAAD	87
00045D6B-ED2E-4F2F-8186-CEE074D875D0	11

COMENTARIOS:

- INSERT INTO each_product_transaction (transaction_id, product_id): Inserta datos en la tabla each_product_transaction.
- SELECT t.id: selecionar el id de tabla transactions (tabla original)

- CAST(j.value AS UNSIGNED) AS product_id: Convierte el valor de cada producto extraído del JSON.(que es texto= (j.value) en número entero.
- JOIN JSON_TABLE(: Convierte un JSON en una tabla virtual.
- **CONCAT('["',:** Enmarca todo con ["..."]
- **REPLACE(t.product_ids, ',', '","'),'"]')**,: reemplaza los valores que están como ‘x,y,z’ dentro de una misma fila a ‘x’, ‘y’ o ‘z’ en diferentes columnas.
- '$[*]': Recorre cada elemento del array JSON.
- COLUMNS (...): Define que cada elemento se llamará value.
- AS j: Alias de la tabla virtual generada por JSON_TABLE.

PASO 4: relacionar la base de datos de la tablas each_product_transactions con las tabla products y transactions a través de 2 FOREIGN KEYS :

Ejercicio 1

Necesitamos conocer el número de veces que se ha vendido cada producto.

PASO 5 - FINAL: Crio una columna temporal llamada ‘sales_number’ para identificar cuántas ventas hay por cada producto y lo ordeno para ver los productos que venden más primero:

```
456 •   SELECT product_id, COUNT(*) AS sales_number
457     FROM each_product_transactions
458     GROUP BY product_id
459     ORDER BY sales_number DESC;
460
```

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

product_id	sales_number
52	2654
29	2635
21	2609
16	2608
66	2601
87	2598
48	2597
33	2597

Entrega:
Almacena en un repositorio de tu GitHub una carpeta que contenga:
Los archivos .sql que contengan todos los scripts.
Una imagen que contenga el diagrama de la base de datos.
Un PDF que contenga una captura de pantalla del workbench donde se pueda observar el script de la consulta que hiciste y el resultado obtenido para cada ejercicio.
En la entrega, coloca el link en el repositorio.

***Descripción*
Partiendo de algunos archivos CSV diseñarás y crearás tu base de datos.
***Importante*
Todas las transformaciones e importaciones necesarias en esta tarea deben realizarse mediante código SQL. NO SE PERMITE utilizar Wizard. Es necesario describir cada paso y realizar una captura de pantalla.

Consultas:
https://www.w3schools.com/sql/sql_having.asp

https://www.w3schools.com/sql/sql_count.asp
https://www.w3schools.com/sql/sql_update.asp
https://www.ibm.com/docs/es/cognos-analytics/11.2.x?topic=relationships-bridge-tables
https://www.linkedin.com/advice/0/how-can-you-use-bridge-table-data-warehousing-q6bre
https://www.youtube.com/watch?v=YYfediyCwAU