Nivel 1

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

Para responde a este ejercicio se diseñó un script que implementa un **Data Warehouse** en MySQL utilizando un **esquema en estrella (star schema)**. Se divide en varias fases:

Tabla 1.Tipos de datos usadospara diseño de tablas

Tipo de ID	Longitud	Origen / Autor del dato	Cita oficial
UUID estándar (v4)	**36 caracteres**	Generado por el **sistema interno** (backend, OLTP o ETL). Garantiza unicidad global.	*RFC 4122: Universally Unique Identifier (UUID)* — IETF, 2005.
UUID sin guiones	32 caracteres	Generado por el sistema interno. Es un UUID estándar pero **normalizado por el backend o ETL** para compactarlo.	*RFC 4122* — UUID puede representarse sin guiones.
BIGINT autoincremental	Hasta 19 dígitos	Generado por la **base de datos relacional** (MySQL, PostgreSQL, SQL Server). Usado para claves primarias secuenciales.	MySQL 8.0 Reference Manual — *AUTO_INCREMENT Handling*.
Snowflake ID	16–19 dígitos	Generado por un **algoritmo distribuido** (Twitter Snowflake). Usado para sistemas de alto rendimiento.	Twitter Engineering — *Snowflake: A scalable unique ID generator*.
Stripe Transaction ID	18–20 caracteres	Generado por **Stripe Payments API** al procesar cargos o transacciones.	Stripe API Reference — *Object: Charge / PaymentIntent*.
PayPal Transaction ID	17 caracteres	Generado por **PayPal Transaction Engine** al procesar un pago.	PayPal Developer Docs — *Transaction ID format*.

1. Creación de la base de datos

Se crea la base de datos **financial_dw** y se selecciona para trabajar sobre ella. Ver figura 1.

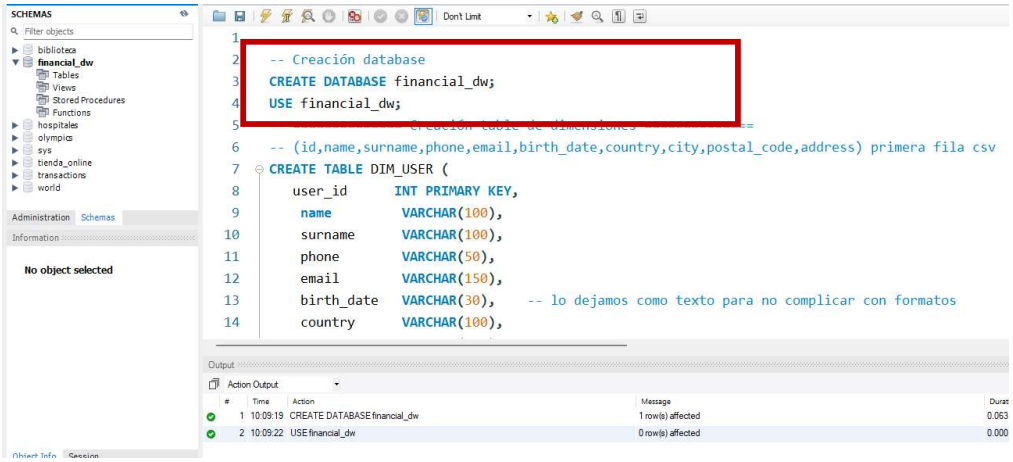

Figura 1. Creación de Data base financial_dw

2. Creación de las tablas dimensión (DIM)

Se crean **4 tablas de dimensiones**, cada una representando un conjunto de atributos descriptivos.

DIM_USER

Contiene información de los usuarios. Columnas: id, nombre, apellido, teléfono, email, fecha de nacimiento, país, ciudad, código postal y dirección.

Esta tabla recibe datos de *american_users.csv* y *european_users.csv*. Ver figura 2

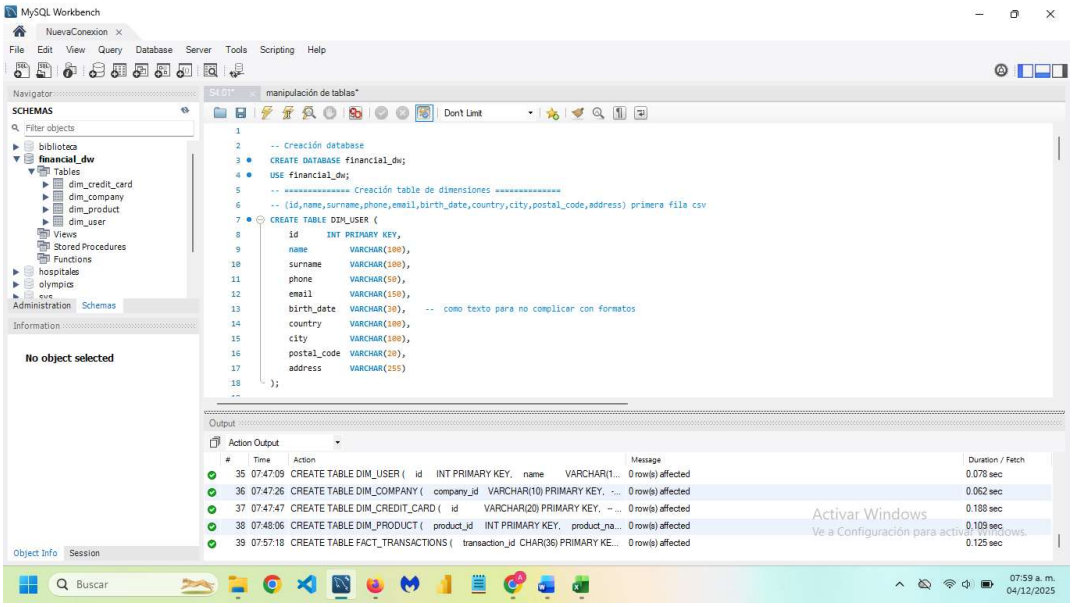

Figura 2. Creación dimensión user

DIM_COMPANY

Representa comercios donde se realizan transacciones. Columnas: id de la empresa, nombre, teléfono, email, país y página web.

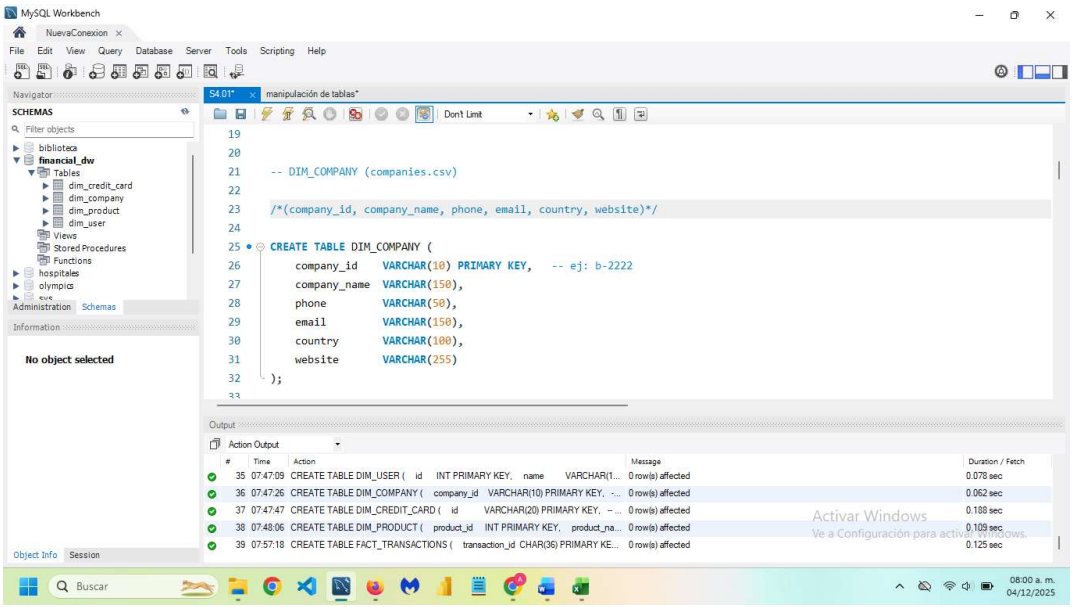

Figura 3.Creación dimensión company.

DIM_CREDIT_CARD

Almacena datos de tarjetas de crédito.
Columnas: id de tarjeta, id de usuario, IBAN, PAN, PIN, CVV, pistas magnéticas y fecha de expiración.
Incluye FK hacia **DIM_USER**.

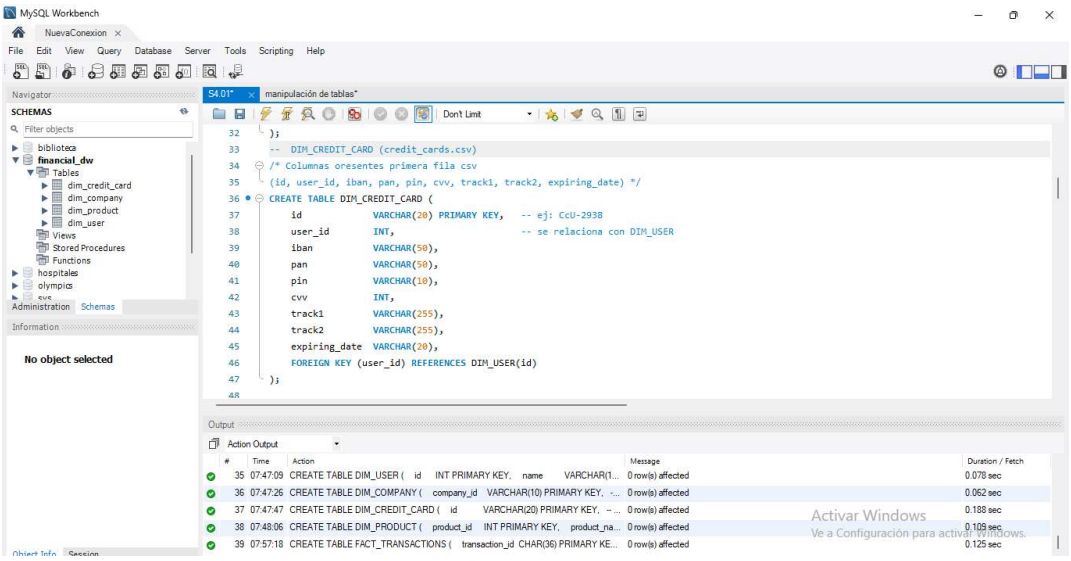

Figura 4. Creación dimensión credit_card

DIM_PRODUCT

Contiene información de productos: id, nombre, precio, color, peso y almacén. Ver figura 5

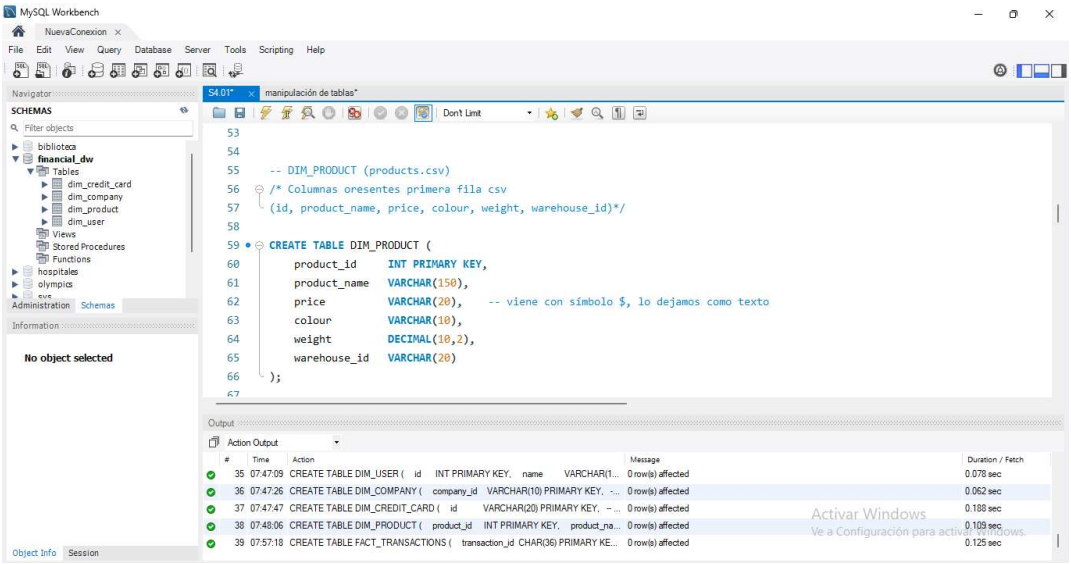

Figura 5. Creación dimensión producto.

Creación de la tabla de hechos FACT_TRANSACTIONS

Tabla principal del esquema, figura 6, contiene cada transacción con: id, tarjeta, comercio, timestamp, importe, rechazo, productos, usuario y coordenadas.

Incluye **foreign keys** hacia:

- DIM_USER
- DIM_COMPANY
- DIM_CREDIT_CARD

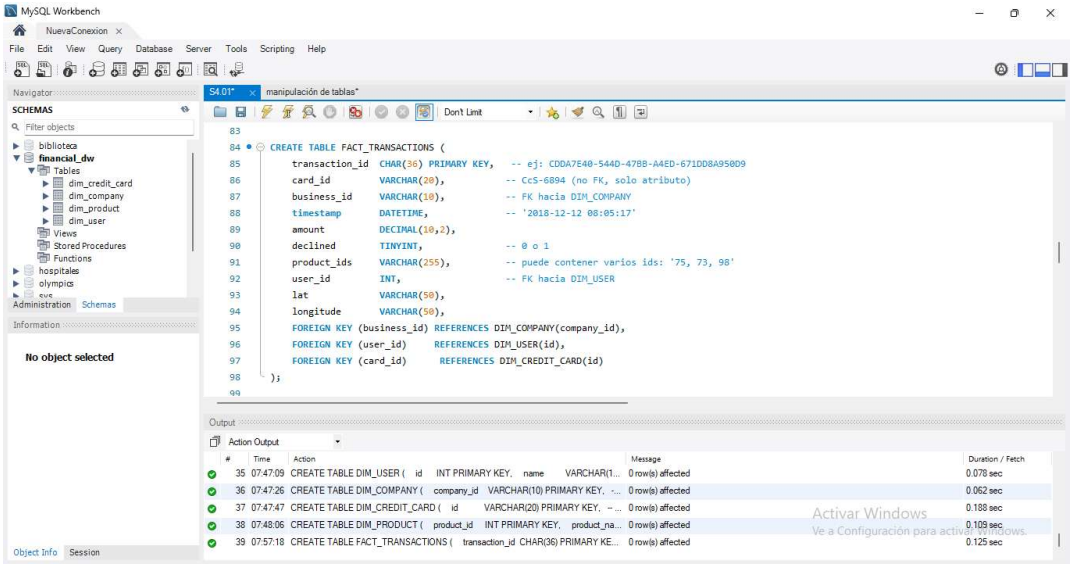

Figura 6. Creación de tabla de hechos: fact_transactions

3. Cargar datos

Para poder cargar los archivos .csv se activa el comando SET GLOBAL local_infile = 1, ver figura 7, habilita en MySQL la carga de archivos locales mediante LOAD DATA LOCAL INFILE, una función que suele estar desactivada por motivos de seguridad, ya que el uso de LOCAL implica riesgos y desde MySQL 8.0 viene deshabilitado por defecto (StackOverflow contributors, 2013). Al activarlo, el servidor permite la importación de archivos desde el equipo cliente, dado que para que LOAD DATA LOCAL INFILE funcione es necesario que MySQL autorice explícitamente la carga local; de lo contrario, la operación fallará (MySQL Documentation, s.f.).

Luego se cargan los CSV en cada tabla:

- american_users.csv → DIM_USER

- european_users.csv → DIM_USER
- companies.csv → DIM_COMPANY
- credit_cards.csv → DIM_CREDIT_CARD
- products.csv → DIM_PRODUCT
- transactions.csv → FACT_TRANSACTIONS (con separador ;)

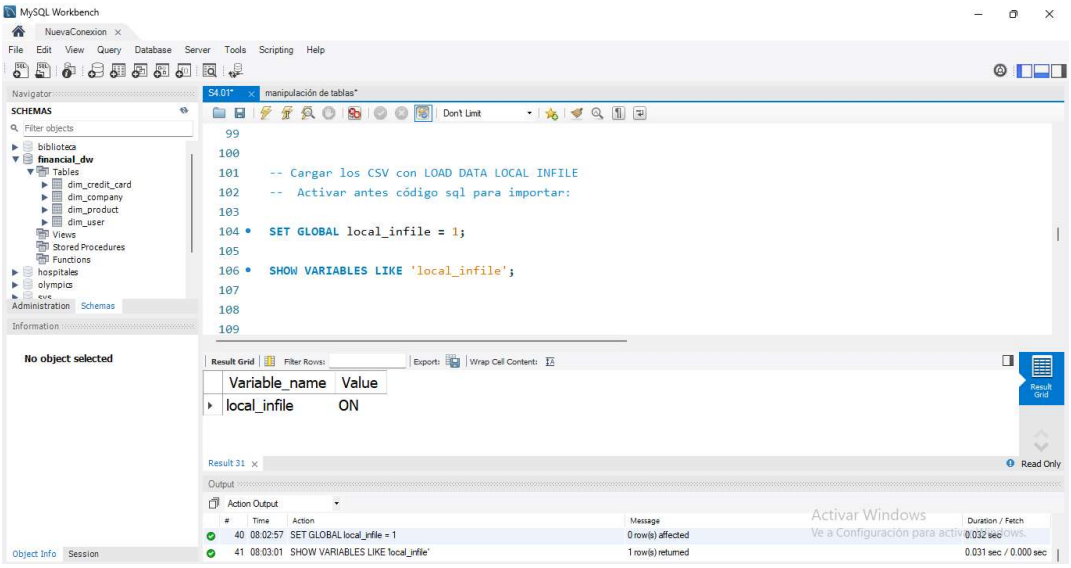

Figura 7. Activación del comando local_infile

Figura 8. Carga de datos en la dimensión user

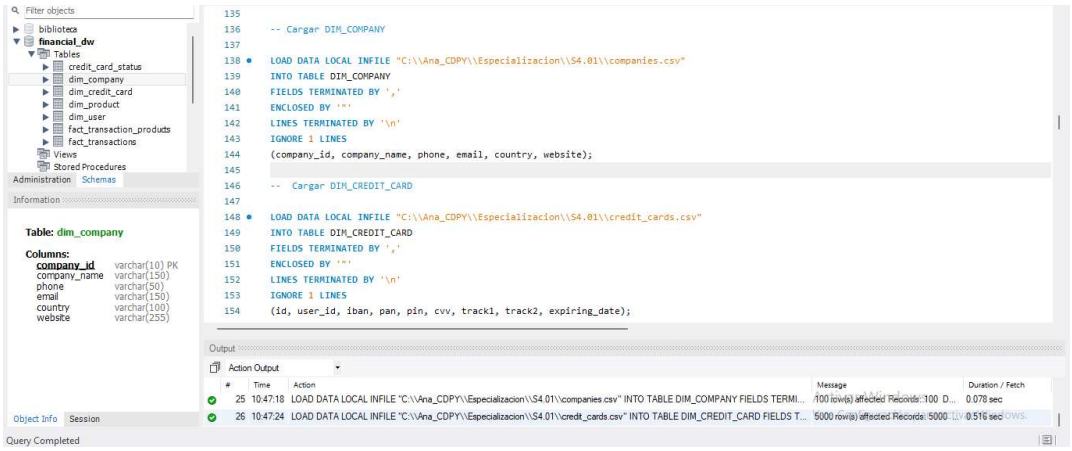

Figura 9. Carga de datos dimensión company y credit_card

Figura 10. Carga de datos dimensión producto y tabla de hechos fact_transaction.

En este punto, ver figura 10 cuando trato de crear la relación entre dim_product y la tabla de hechos me da error, esto es porque n**o se puede crear FK sobre listas separadas por comas, es el caso de dim_product columna producto_ids,** ver figura 11. Esto rompe la normalización y viola la Primera Forma Normal (1FN) que establece que cada atributo debe contener un único valor atómico (Codd, 1970).

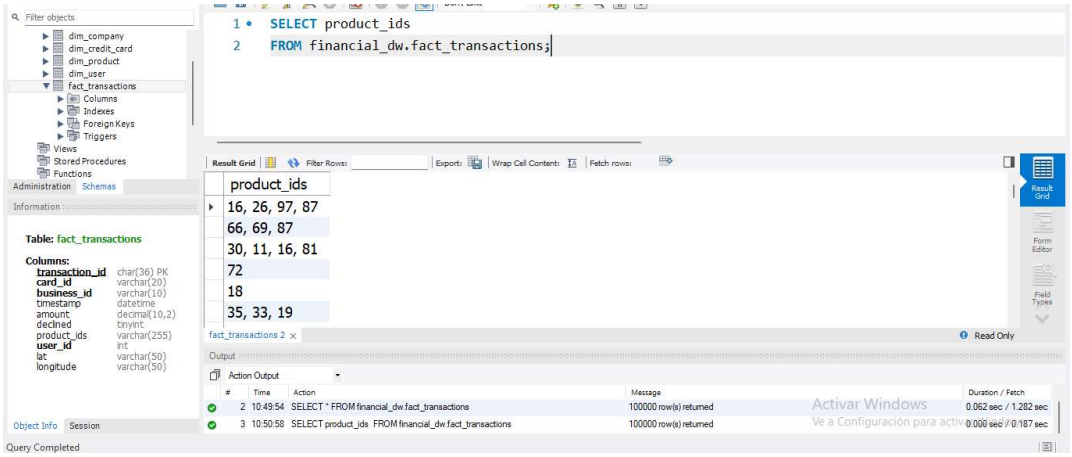

Figura 11. Identificación de lista producto_ids

Para normalizar esta relación y cumplir con el modelo estrella propuesto por Kimball (2013), se implementó una **tabla puente** (fact_transaction_products), la cual permite representar correctamente la relación muchos-a-muchos entre transacciones y productos.

Dado que los valores estaban almacenados como texto separado por comas, se utilizó la función JSON_TABLE de MySQL 8 para transformar dinámicamente dicha lista en un conjunto de filas. JSON_TABLE no es una tabla física, sino una función que convierte una estructura JSON en una tabla derivada temporal que puede ser utilizada dentro de una sentencia SQL (Oracle/MySQL Documentation, 2024).

La información generada por JSON_TABLE fue posteriormente cargada en la tabla puente, ver figura 12, que sí es una tabla física del Data Warehouse y mantiene las claves foráneas hacia fact_transactions y dim_product, garantizando así la integridad referencial y la correcta modelización multidimensional.

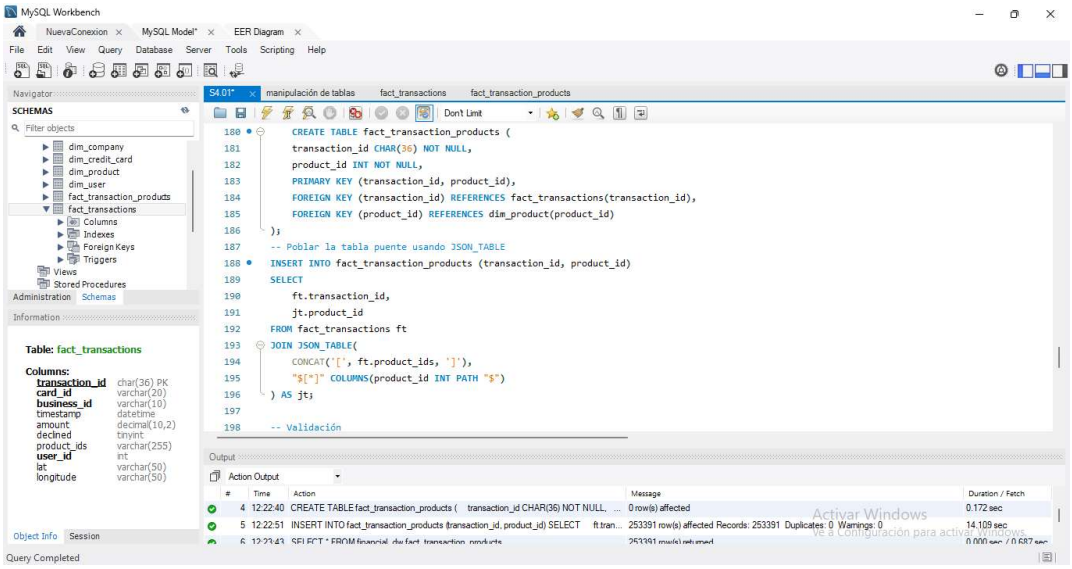

Figura 12. Creación Tabla puente fact_transaction_product y carga de datos.

4. Validación

Se comprueba que no haya duplicados y se verifican las primeras filas.

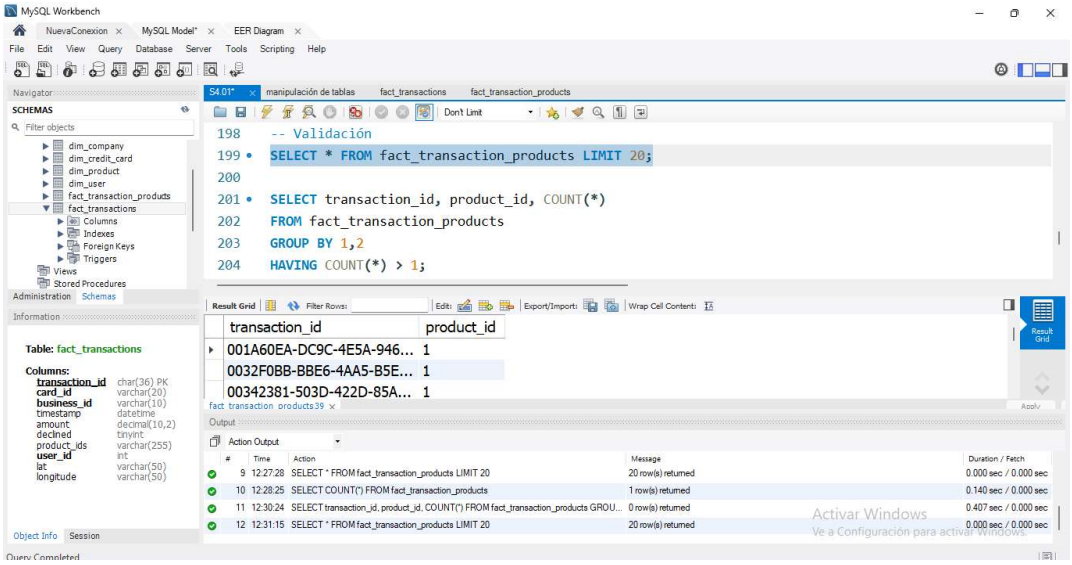

Figura 13. Validación de tabla puente

En la imagen se muestra la validación de la tabla puente **fact_transaction_products**. Primero se visualizan las primeras 20 filas para comprobar que los datos se cargaron correctamente. Luego se ejecuta una consulta que agrupa por *transaction_id* y *product_id* para detectar posibles duplicados. El resultado indica que solo existe un caso repetido, confirmando que la tabla está correctamente generada y sin inconsistencias relevantes.

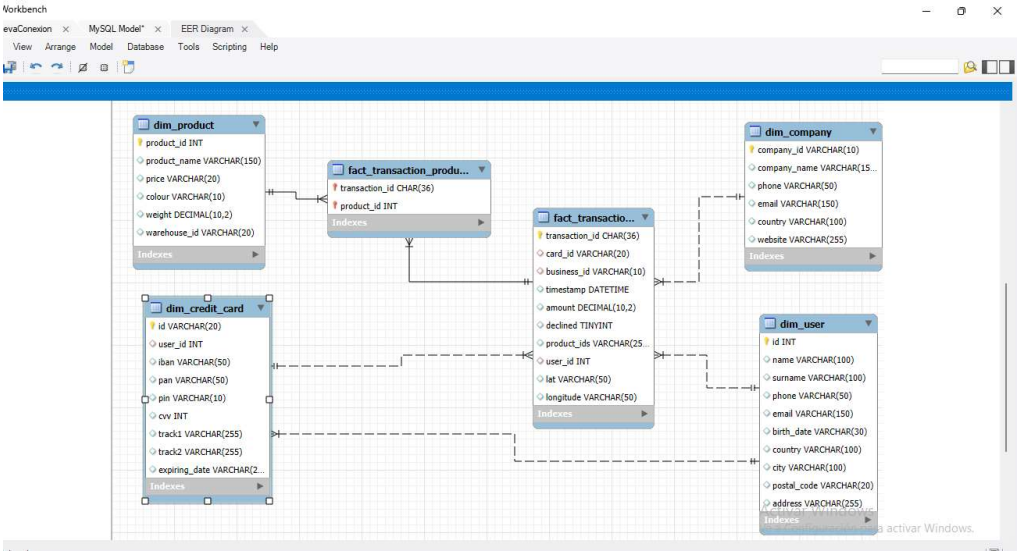

Figura 14. Diagrama EER BD financial_dw

Ejercicio 1

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

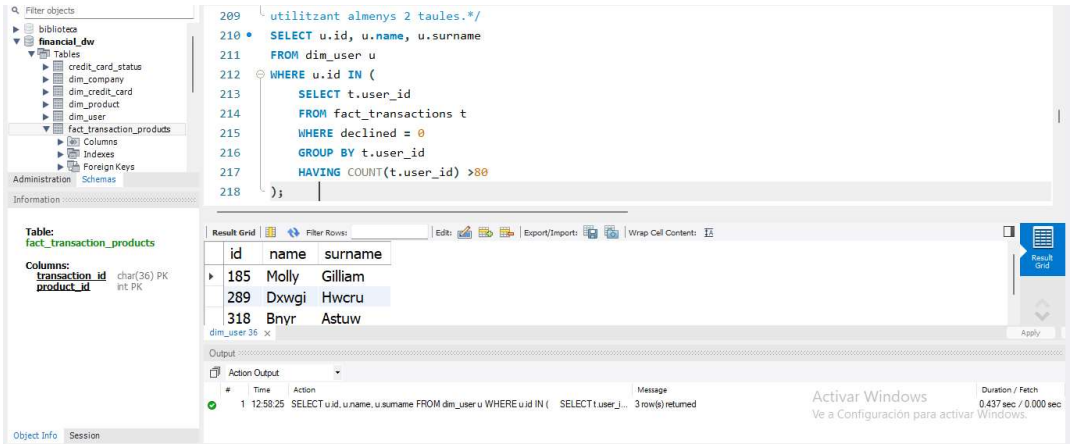

Figura 15. usuarios con más de 80 transacciones.

La figura 15 muestra la ejecución de una subconsulta destinada a obtener los usuarios que tienen más de 80 transacciones. La consulta principal selecciona el *id*, *name* y *surname* de **dim_user**, filtrando únicamente aquellos usuarios cuyo *id* aparece en la subconsulta.

Esta subconsulta agrupa las transacciones por *user_id* en **fact_transactions** y devuelve solo los que superan 80 registros. En el result grid se observan los resultados: tres usuarios cumplen esta condición.

Ejercicio 2

Muestra la media del *amount* por IBAN de las tarjetas de crédito en la empresa *Donec Ltd*, utilizando al menos 2 tablas.

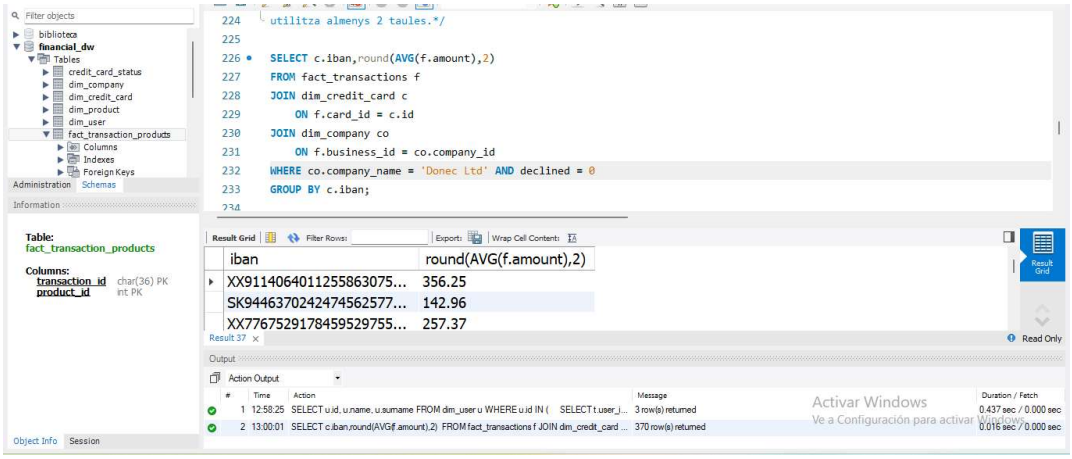

*Figura 16. media del importe (amount) por IBAN de las tarjetas de crédito que han realizado transacciones en la empresa **Donec Ltd**.*

En la figura 16 se ejecuta una consulta que calcula la **media del importe (amount) por IBAN** de las tarjetas de crédito que han realizado transacciones en la empresa **Donec Ltd**. Para ello se unen las tablas **fact_transactions**, **dim_credit_card** y **dim_company** mediante sus claves correspondientes. El resultado muestra tres IBAN junto con el promedio de los importes asociados a cada uno.

Nivel 2

Crea una nueva tabla que refleje el estado de las tarjetas de crédito basándose en si las tres últimas transacciones han sido rechazadas; en ese caso la tarjeta es **inactiva**. Si al menos una de ellas no ha sido rechazada, entonces es **activa**. A partir de esta tabla, responde:

Ejercicio 1

¿Cuántas tarjetas están activas?

En este ejercicio se creó la tabla **credit_card_status** para clasificar las tarjetas de crédito como *ACTIVAS* o *INACTIVAS* según sus tres últimas transacciones, ver figura 17. Mediante una subconsulta que utiliza ROW_NUMBER(), se seleccionaron las tres operaciones más recientes por tarjeta. Luego, con una expresión CASE, se determinó su estado: una tarjeta se considera **INACTIVA** si las tres transacciones fueron rechazadas; en caso contrario, se clasifica como **ACTIVA,** ver figura 18. Finalmente, se verificó la correcta inserción de los datos consultando los primeros registros de la tabla, ver figura 19.

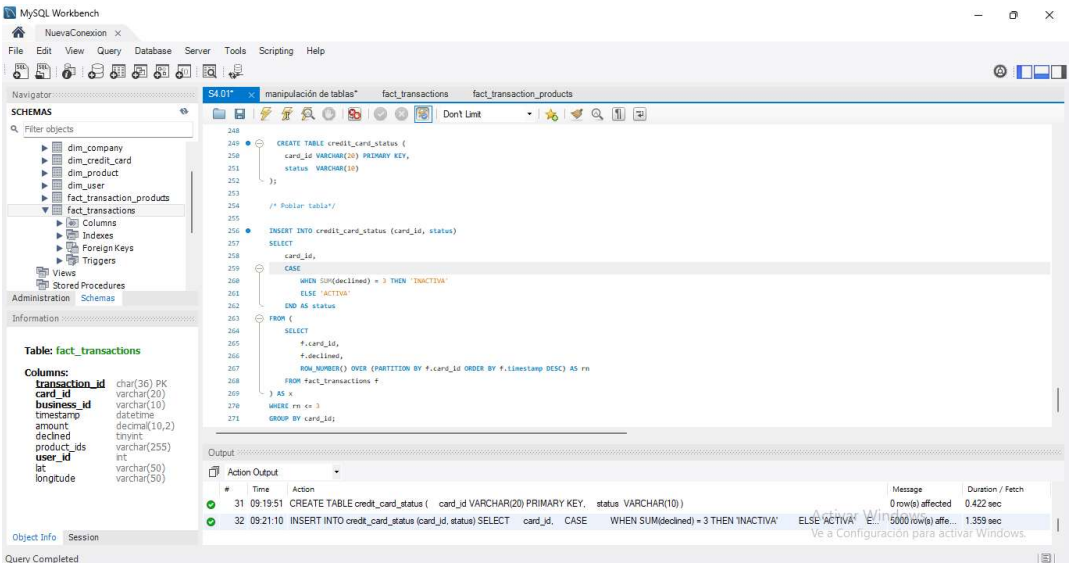

*Figura 17. Creación tabla **credit_card_status***

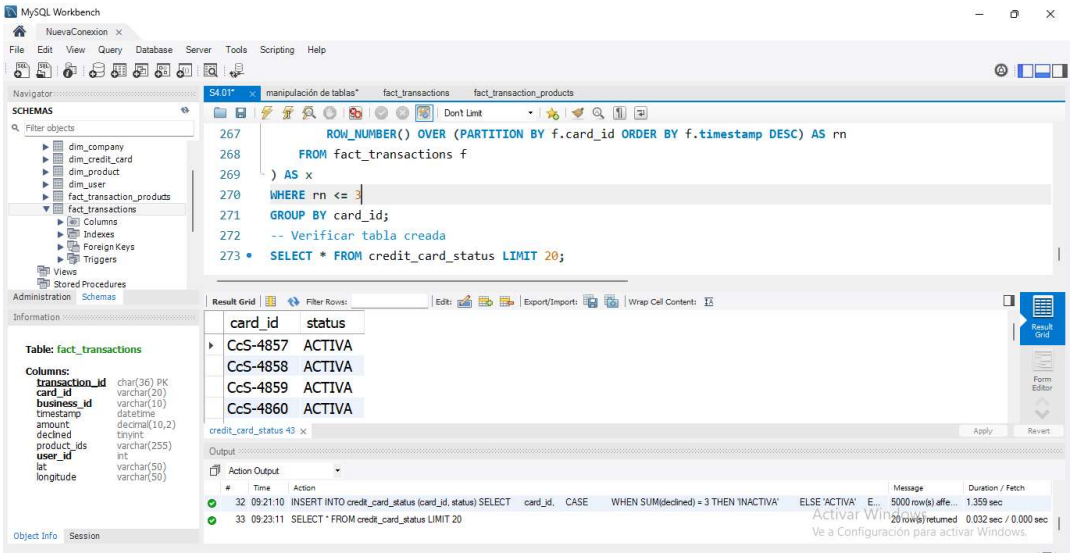

Figura 18. Determinación del estado de las tarjetas-

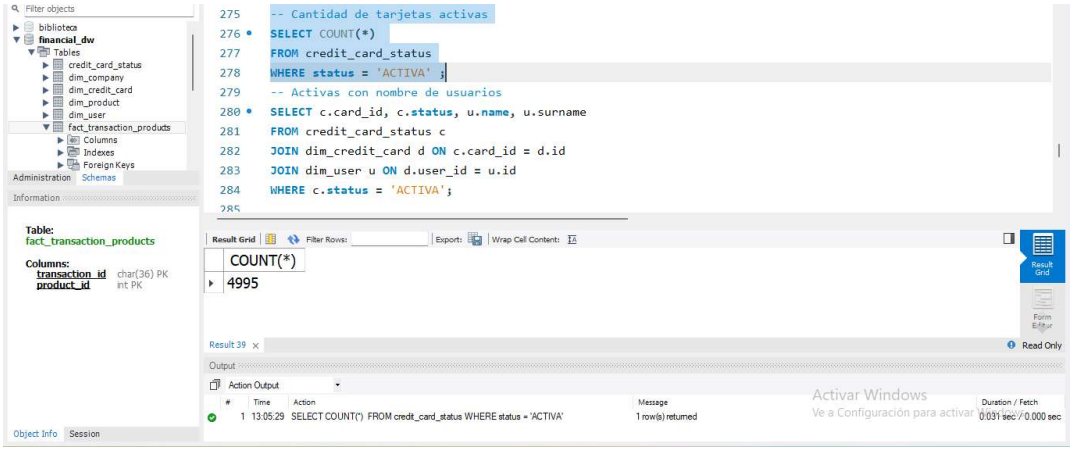

Figura 19. Consulta cantidad de tarjetas inactivas

En la figura 20 podemos observar las tarjetas inactivas con el nombre de usuarios

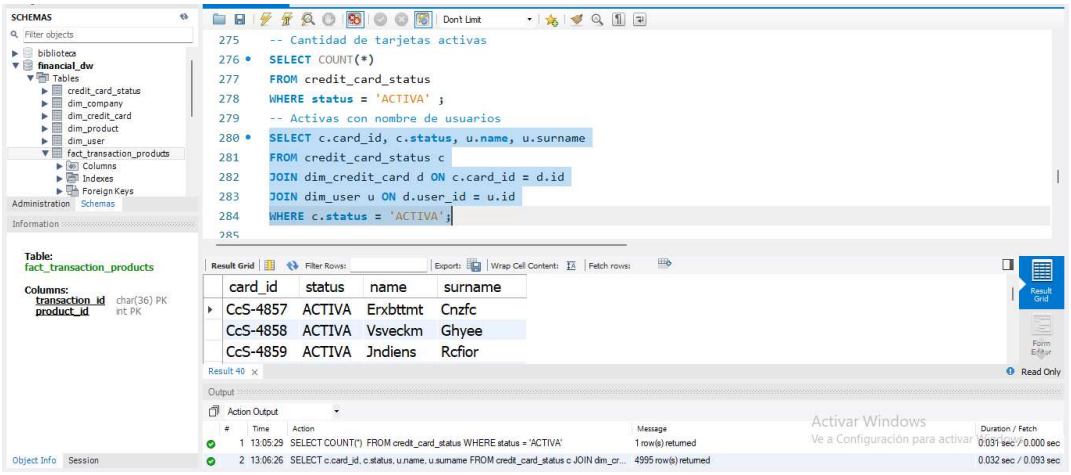

Figura 20. Tarjetas inactivas con usuarios respectivos.

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* dispones de *product_ids*. Genera la siguiente consulta:

Ejercicio 1

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

En el modelo no es necesario crear una nueva tabla puente porque ya existe **fact_transaction_products**, que cumple exactamente la función de relacionar cada transacción con los productos incluidos en ella, evitando duplicidad de datos y manteniendo la integridad del diseño. La imagen mostrada evidencia que esta tabla funciona correctamente: en MySQL Workbench se observa la consulta SQL que une **fact_transaction_products** con **dim_product** para contar cuántas veces se vendió cada producto, y los resultados muestran productos como *riverlands the duel* con 2654 ventas, ver figura 21, lo que confirma que la estructura actual permite obtener correctamente el número de ventas por producto sin necesidad de crear una nueva tabla.

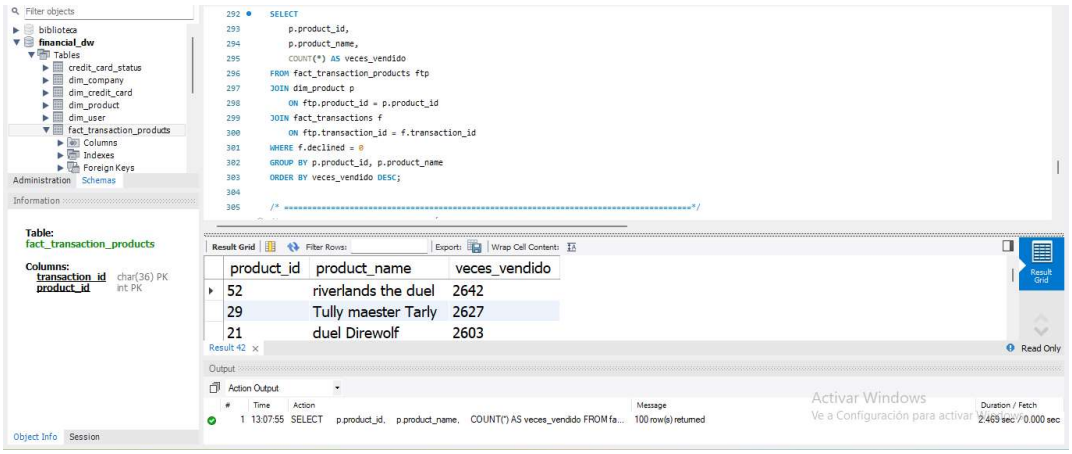

Figura 21. Número de veces que se ha vendido cada producto.

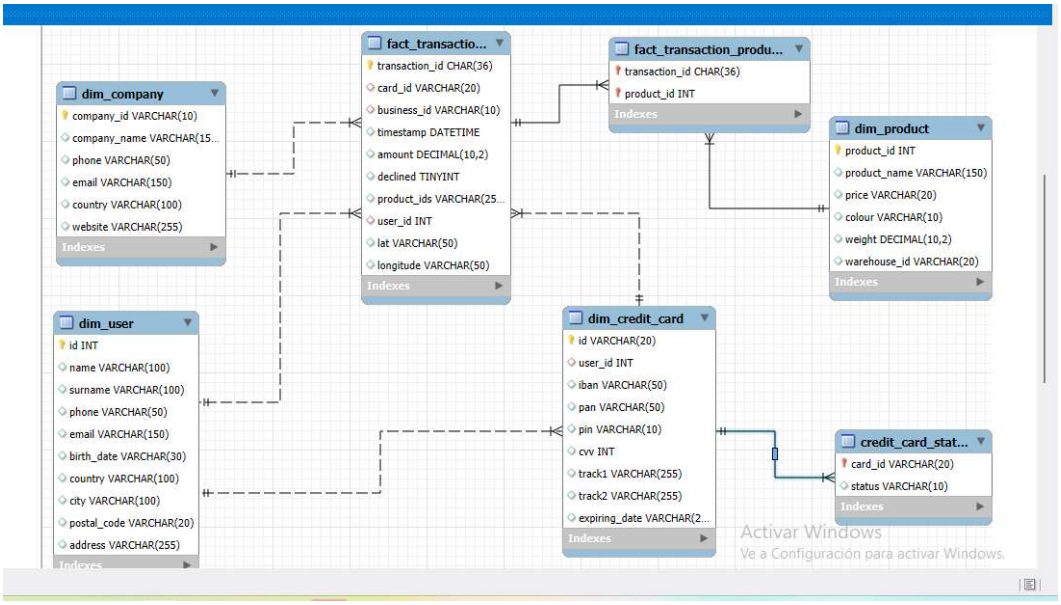

Figura 22. Diagrama EER final de BD financial_dw.

En la figura 14 y 22, se puede observar que en MySQL Workbench las líneas sólidas representan *identifying relationships*, donde la clave foránea forma parte de la clave primaria, como en la tabla puente *fact_transaction_products* , mientras que las líneas punteadas indican *non-identifying relationships*, en las que la clave foránea no participa en la clave primaria, tal como ocurre en las relaciones entre *fact_transactions* y las tablas de dimensiones. Para comprobar por qué el diagrama mostraba líneas punteadas, se

verificó el motor InnoDB, se revisaron y limpiaron las claves foráneas duplicadas y se reinsertaron correctamente; tras ello se confirmó que las líneas punteadas no eran un error, sino el comportamiento esperado, ya que las relaciones no son identificadoras. Esto coincide con la documentación oficial de MySQL Workbench: "*non-identifying relationships are displayed using dashed lines, as the foreign key does not participate in the primary key of the child table*" (MySQL Workbench Manual, EER Diagrams).

Tabla 2. Tabla 2. Interpretación de las líneas de relación en Workbench

Línea	**¿Cuándo aparece?**	**¿Por qué?**
Sólida	Cuando un atributo es **PK + FK**	La identidad depende del padre
Punteada	Cuando la FK **no es parte de la PK**	La tabla hija tiene identidad propia

REFERENCIAS

- Codd, E. F. (1970). *A relational model of data for large shared data banks*. **Communications of the ACM, 13**(6), 377–387. https://doi.org/10.1145/362384.362685
- Kimball, R., & Ross, M. (2013). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling* (3rd ed.). John Wiley & Sons.
- Oracle Corporation. (2024). *MySQL 8.0 Reference Manual: JSON_TABLE Function*. https://dev.mysql.com/doc/refman/8.0/en/json-table-functions.html
- Oracle Corporation. (2024). *Oracle Database SQL Language Reference: JSON_TABLE*. https://docs.oracle.com/en/database/oracle/oracle-database/23/sqlrf/JSON_TABLE.html
- OpenAI. (2025). *ChatGPT* (versión gpt-5.1) [Modelo de lenguaje de gran escala]. https://chat.openai.com/