

Informe Técnico

Normalización a Tercera Forma Normal (3FN) y Análisis de Performance en SQL Server

Proyecto: LegacyRetail

Diplomado: Gestión de Datos – 2026

Estudiante: Karen Suarez

1. Introducción

Durante varios años, LegacyRetail S.A. almacenó la información de sus ventas en un archivo plano sin ningún tipo de estructura relacional. Este archivo contenía todos los datos mezclados en una sola tabla, lo que dificultaba el análisis, generaba datos repetidos y provocaba errores al momento de realizar consultas.

Al intentar migrar esta información directamente a SQL Server, se evidenciaron múltiples problemas tanto a nivel de diseño como de rendimiento. Por esta razón, el objetivo de este proyecto fue reorganizar la información aplicando los principios de la **Tercera Forma Normal (3FN)** y analizar cómo un buen diseño impacta directamente en el performance de las consultas SQL.

El trabajo se divide en dos partes principales:

- **Misión A:** Normalización del modelo de datos.
 - **Misión B:** Análisis de rendimiento y comparación de consultas.
-

2. Misión A — Normalización del Modelo de Datos

2.1 Análisis inicial del problema

El primer problema identificado fue que toda la información se encontraba en una sola estructura, donde se repetían constantemente los datos de clientes, productos y sucursales. Por ejemplo, un mismo cliente aparecía varias veces con el mismo correo electrónico, y una misma sucursal se repetía en múltiples registros de ventas.

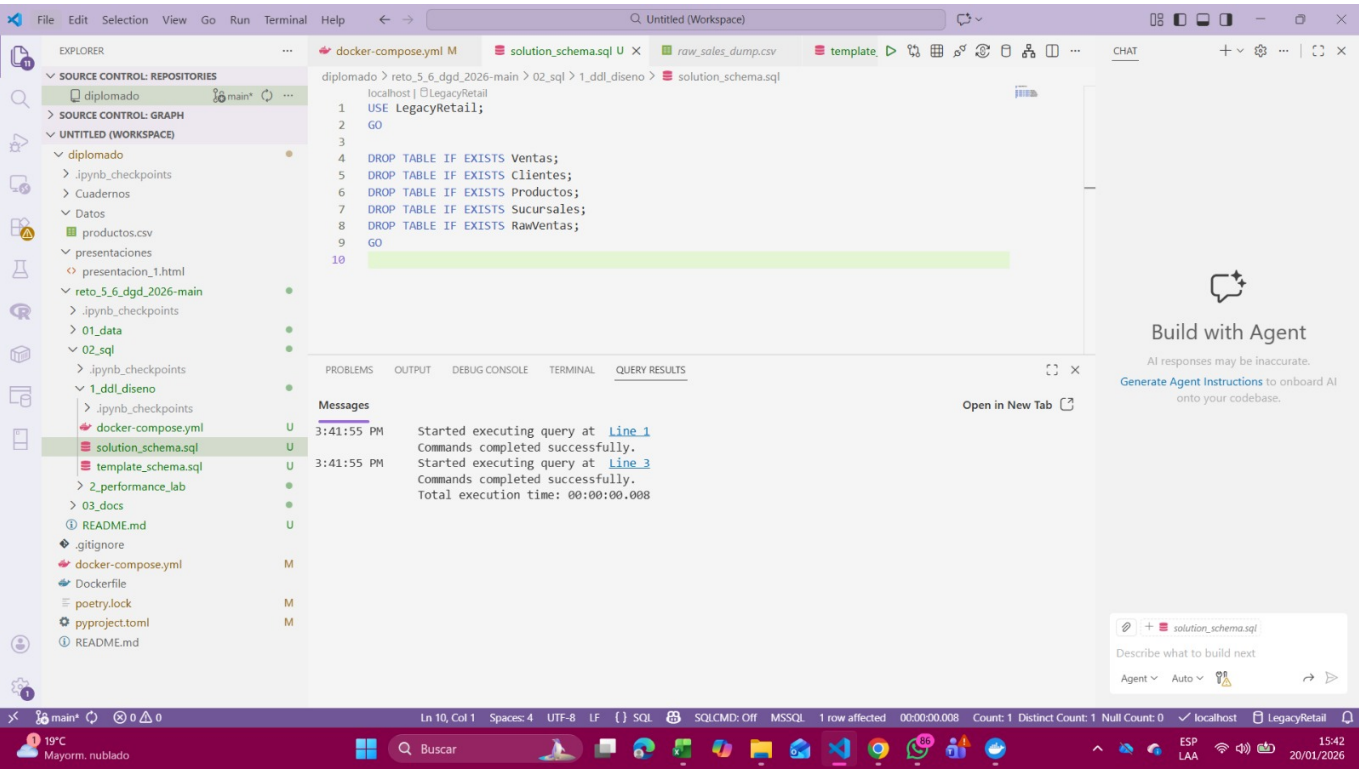
Esto generaba:

- Redundancia de información
- Riesgo de inconsistencias
- Dificultad para mantener los datos actualizados

Antes de normalizar, fue necesario entender qué información correspondía realmente a cada entidad del negocio.

2.2 Limpieza del entorno de trabajo

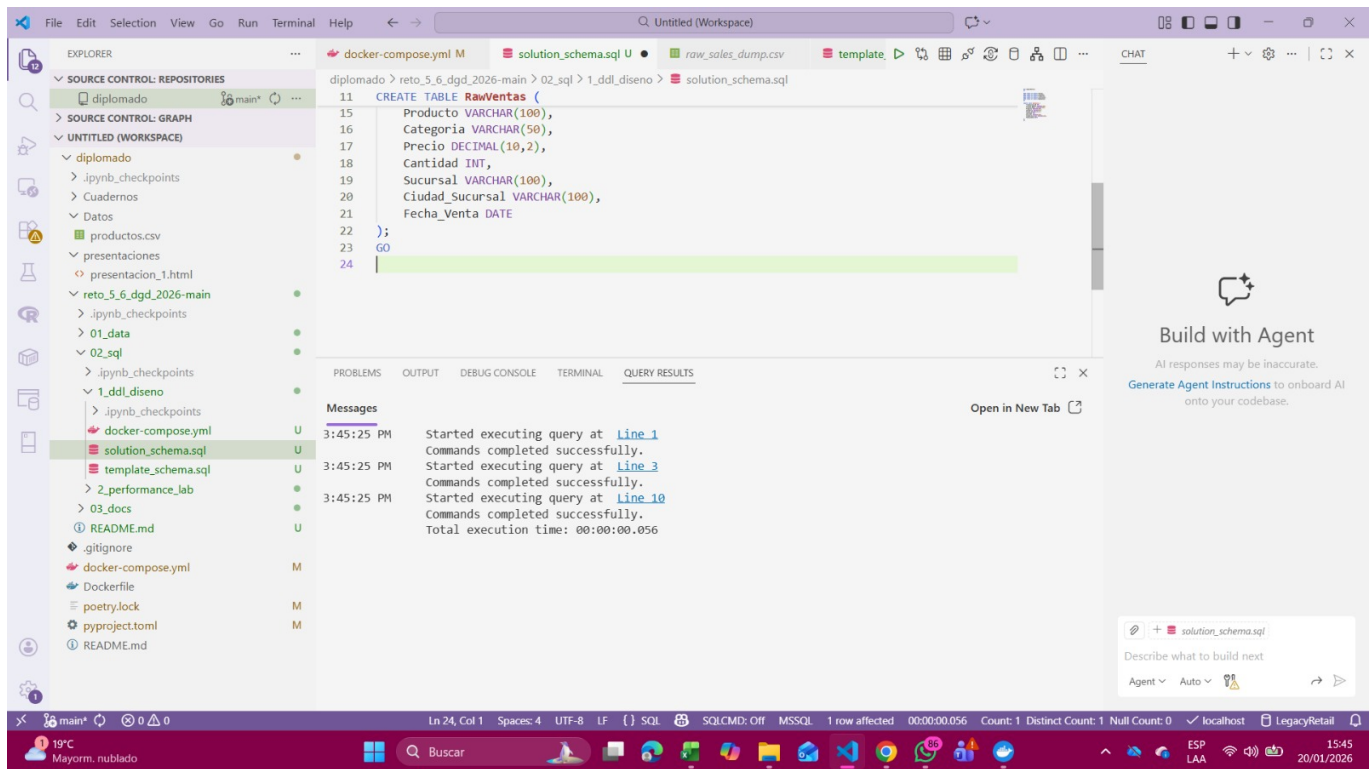
Antes de comenzar con el diseño del modelo relacional, se realizó una limpieza completa del entorno de trabajo, eliminando todas las tablas existentes y asegurando que la base de datos LegacyRetail estuviera lista para una nueva implementación.



Este paso fue clave para evitar errores causados por ejecuciones previas y para poder validar correctamente cada etapa del proceso.

2.3 Creación de la tabla de staging (RawVentas)

Se creó la tabla RawVentas como una tabla de staging, cuyo objetivo fue almacenar los datos originales del archivo CSV sin aplicar transformaciones inmediatas.



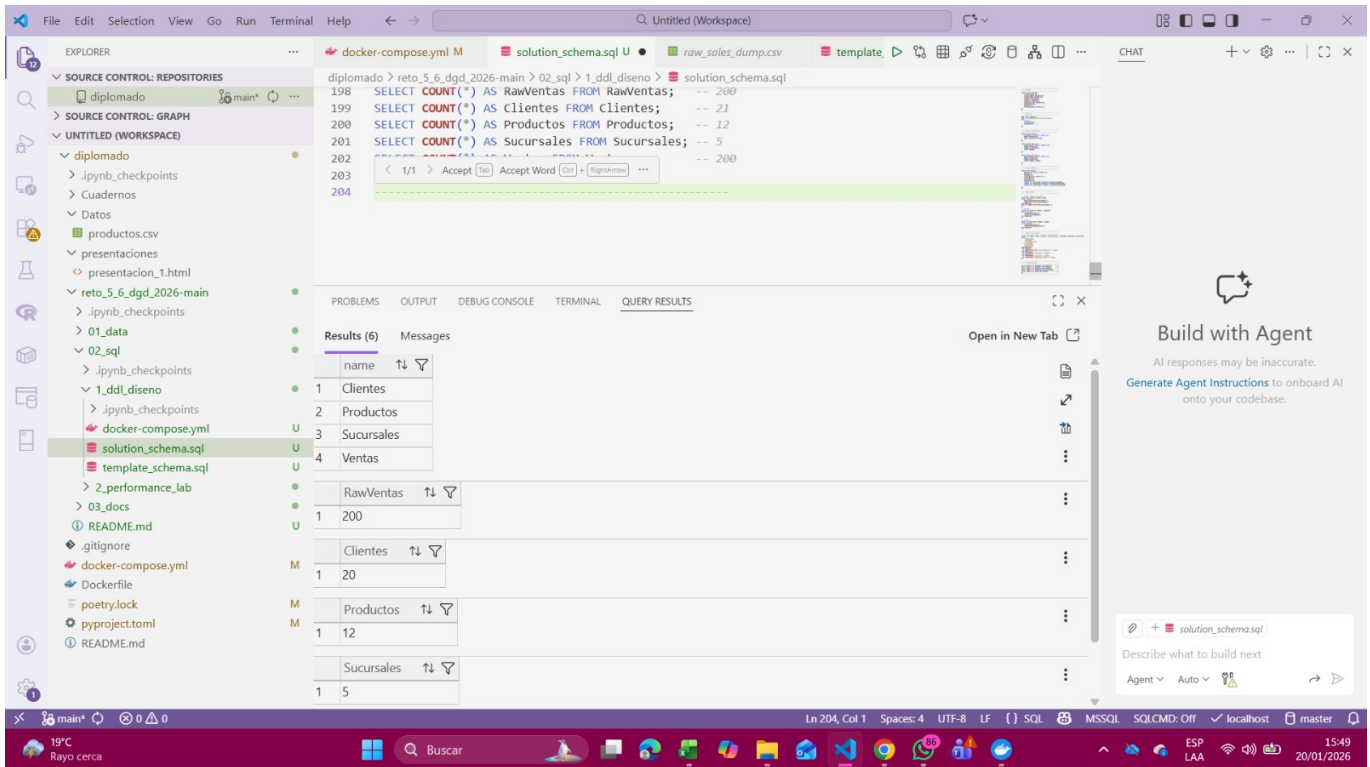
Durante esta etapa se identificaron varios problemas en los datos, como espacios en blanco, diferencias en el formato de texto y repetición de información, lo cual confirmó la necesidad de normalizar el modelo.

2.4 Diseño y creación de las tablas normalizadas

Luego del análisis de los datos crudos, se definieron las entidades principales del sistema y se crearon las siguientes tablas:

- Clientes
- Productos
- Sucursales
- Ventas

Cada tabla fue diseñada con una clave primaria y con atributos que dependen únicamente de dicha clave, cumpliendo con los principios de la Tercera Forma Normal.



The screenshot shows a Visual Studio Code editor with a workspace named 'Untitled (Workspace)'. The Explorer panel on the left shows a project structure with folders like 'diplomado', 'Cuadernos', 'Datos', 'productos.csv', 'presentaciones', 'presentacion_1.html', 'reto_5_6_dgd_2026-main', '01_data', '02_sql', '03_docs', 'README.md', '.gitignore', 'docker-compose.yml', 'Dockerfile', 'poetry.lock', 'pyproject.toml', and 'README.md'. The main editor area shows a SQL query in 'solution_schema.sql' with the following content:

```
198 SELECT COUNT(*) AS RawVentas FROM RawVentas; -- 200
199 SELECT COUNT(*) AS Clientes FROM Clientes; -- 21
200 SELECT COUNT(*) AS Productos FROM Productos; -- 12
201 SELECT COUNT(*) AS Sucursales FROM Sucursales; -- 5
202
203
204
```

The Results (6) Messages panel shows the following data:

name	count
1 Clientes	21
2 Productos	12
3 Sucursales	5
4 Ventas	200

The bottom status bar shows the current file is 'solution_schema.sql' at line 204, column 1, with a UTF-8 encoding and LF line endings. The bottom right corner shows the system tray with the date 20/01/2026 and time 15:49.

La tabla **Ventas** se definió como la tabla central del modelo, encargada de relacionar a los clientes, productos y sucursales.

2.5 Carga de datos y dificultades encontradas

Durante la carga de datos hacia las tablas normalizadas se presentaron algunos retos, especialmente relacionados con la identificación única de los clientes y la eliminación de duplicados.

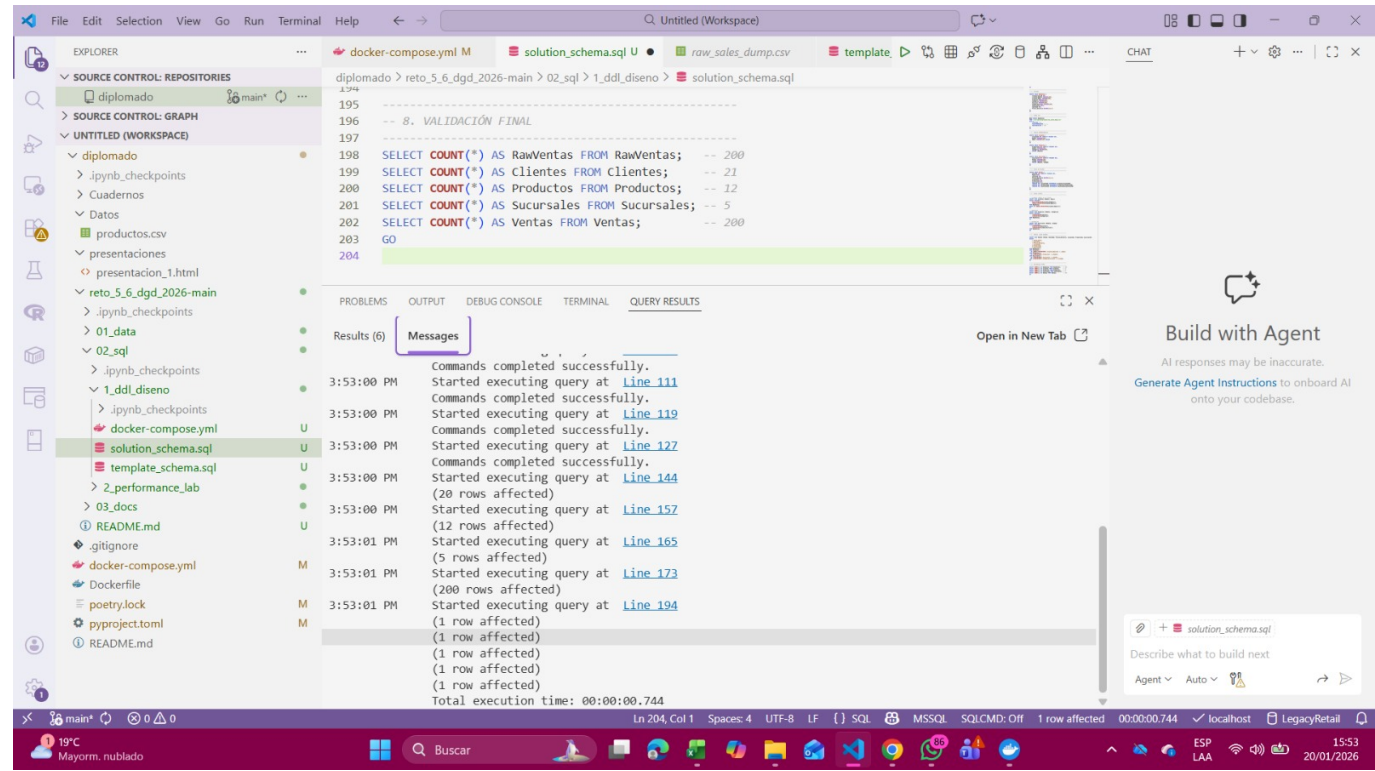
Para resolver esto, se utilizó el correo electrónico como identificador único de cliente, permitiendo consolidar múltiples registros repetidos en una sola fila dentro de la tabla **Clientes**.

2.6 Validación de registros cargados

Una vez completada la carga, se realizó un conteo de registros para validar que la información se hubiera distribuido correctamente entre las tablas.

Resultados obtenidos:

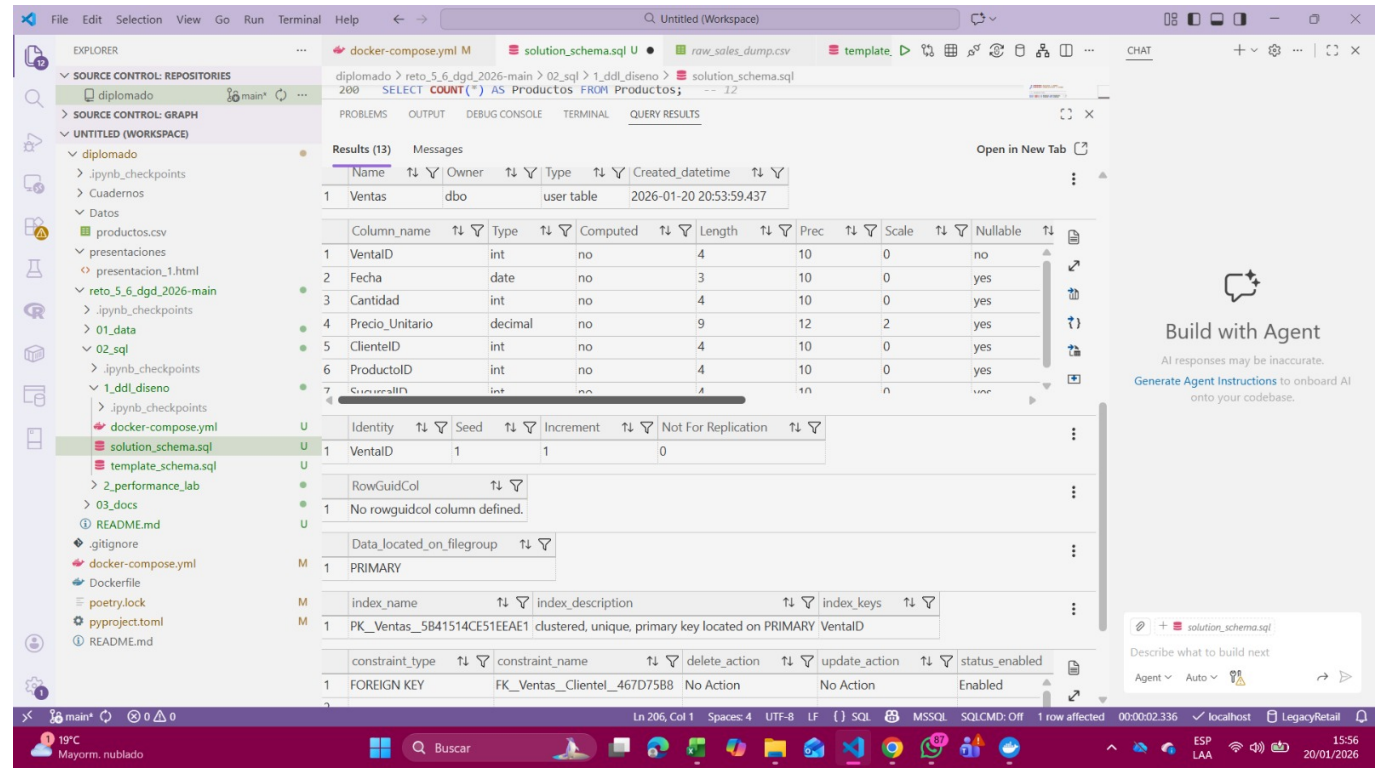
- RawVentas: 200 registros
- Clientes: 21 registros
- Productos: 12 registros
- Sucursales: 5 registros
- Ventas: 200 registros



Estos resultados confirmaron que el proceso de normalización fue exitoso y que la redundancia fue eliminada.

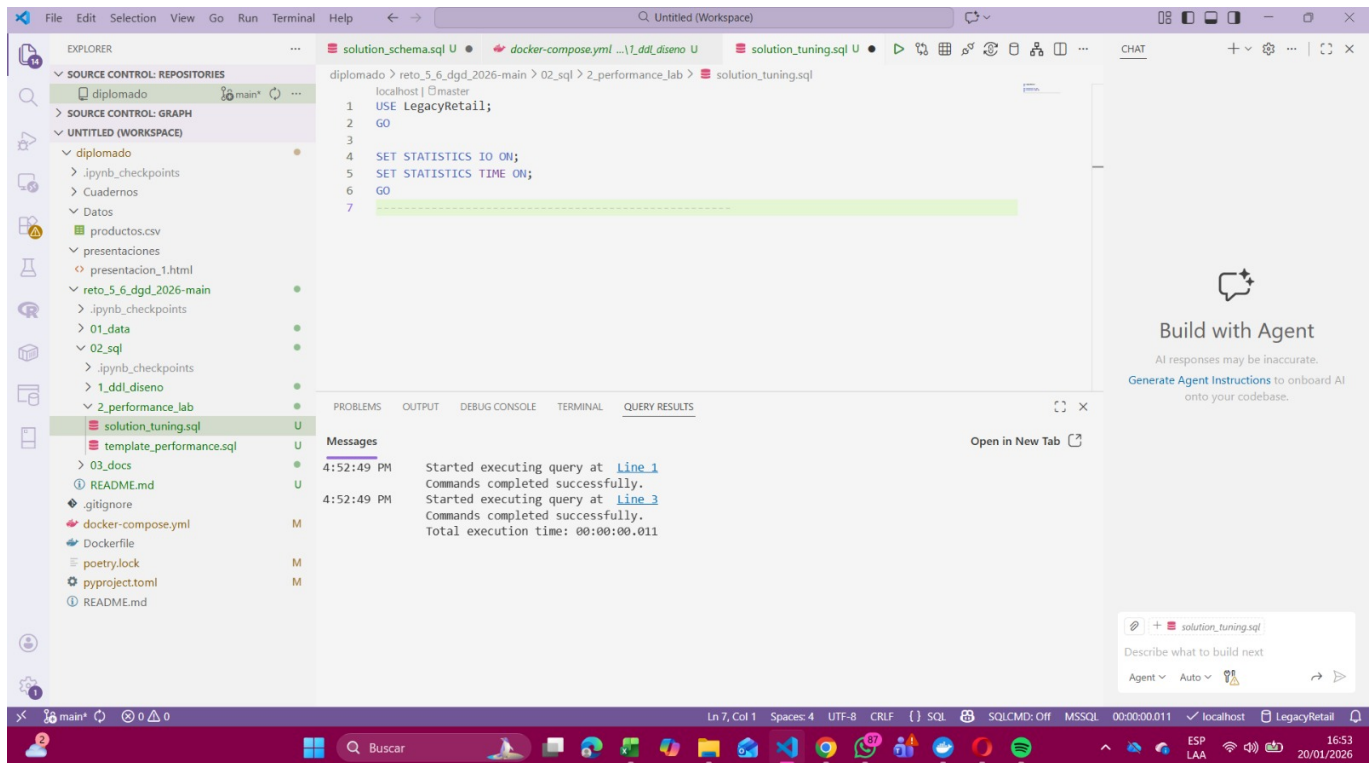
2.7 Revisión de la estructura de la tabla Ventas

Se revisó la estructura de la tabla **Ventas** para verificar que los tipos de datos, la clave primaria y las columnas de referencia estuvieran correctamente definidas.



2.8 Validación de integridad referencial

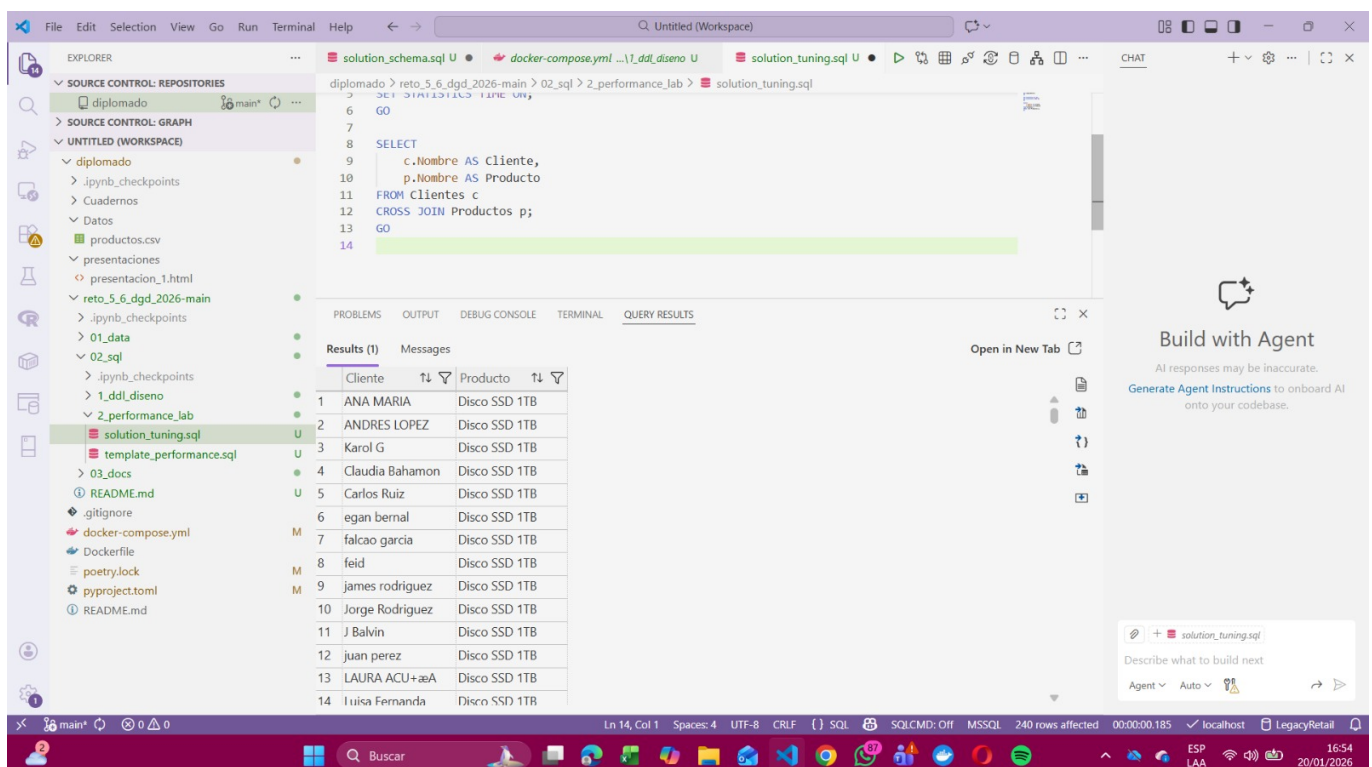
Finalmente, se validó que las claves foráneas estuvieran correctamente configuradas, asegurando la relación entre las tablas y evitando la inserción de datos inconsistentes.



3. Misión B — Análisis de Performance

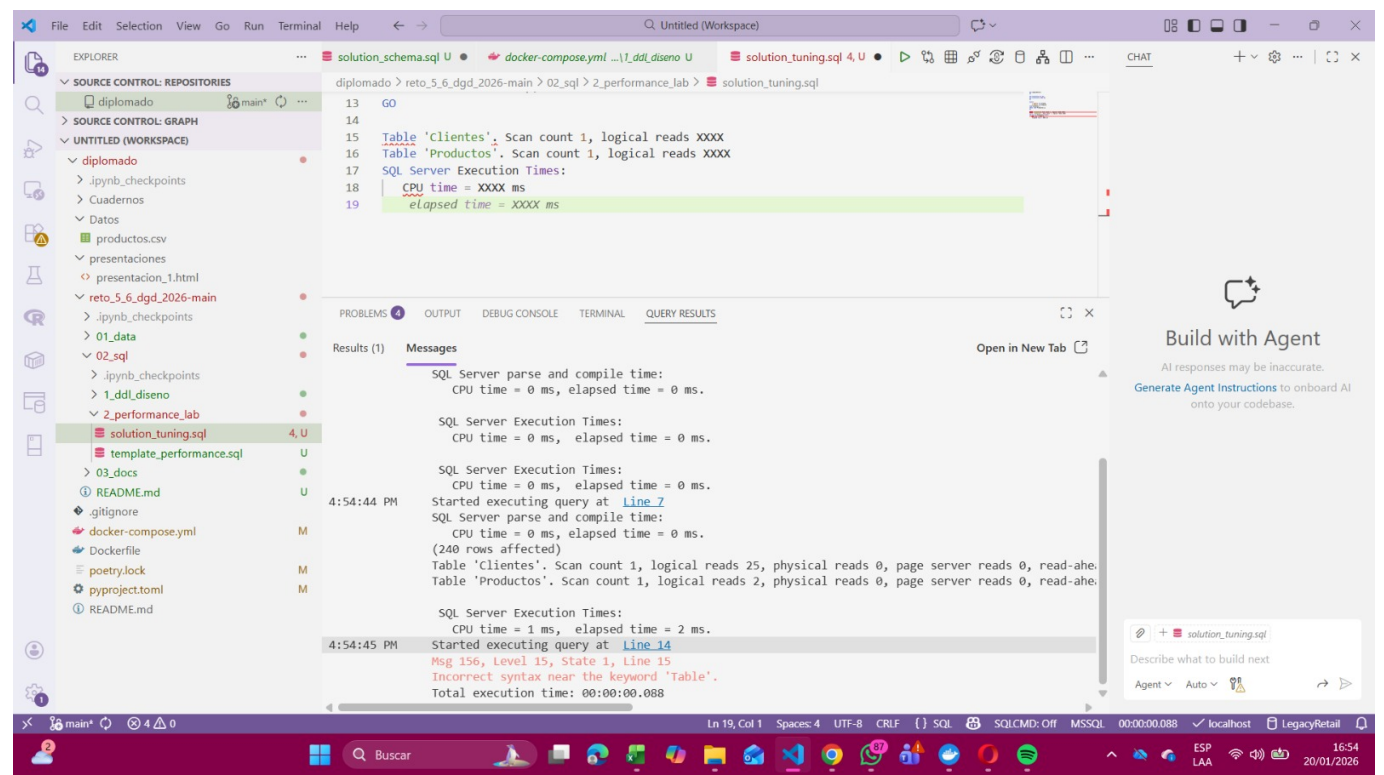
3.1 Preparación del entorno de pruebas

Para analizar el rendimiento de las consultas, se activaron las opciones **STATISTICS IO** y **STATISTICS TIME**, lo que permitió observar el consumo de recursos de cada consulta ejecutada.



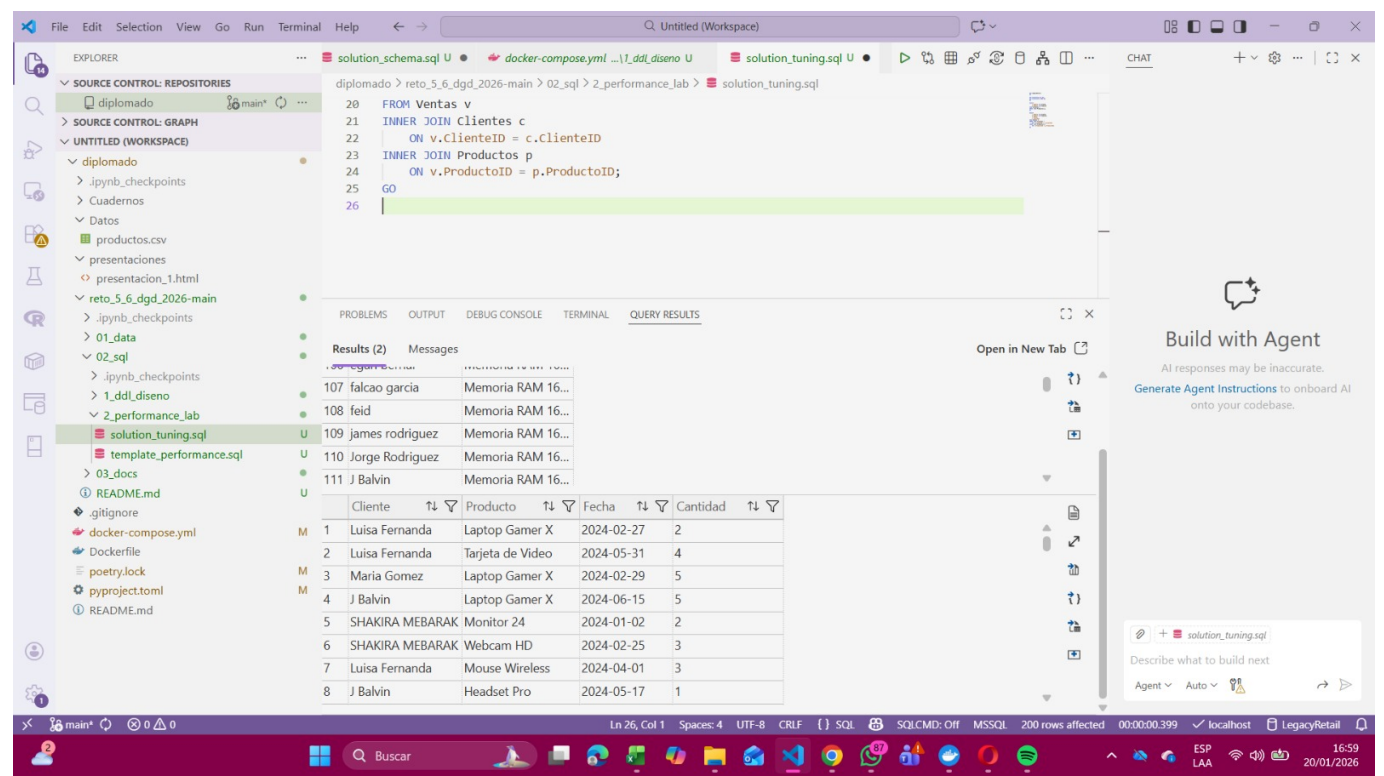
3.2 Consulta con CROSS JOIN y problemas detectados

Se ejecutó una consulta utilizando **CROSS JOIN** entre las tablas **Clientes** y **Productos**. Esta consulta generó un producto cartesiano, combinando todos los registros de ambas tablas.



3.3 Resultados y métricas del CROSS JOIN

El resultado de esta consulta generó una gran cantidad de filas sin relación real de negocio, lo que se reflejó directamente en un aumento del consumo de recursos.



The screenshot shows a Visual Studio Code workspace with the following components:

- Explorer:** Displays a file tree with folders like 'SOURCE CONTROL: REPOSITORIES', 'SOURCE CONTROL: GRAPH', and 'UNTITLED (WORKSPACE)'. The file 'solution_tuning.sql' is selected.
- Editor:** Shows the SQL query in 'solution_tuning.sql':

```
diplomado > reto_5_6_dgd_2026-main > 02_sql > 2_performance_lab > solution_tuning.sql
31 GO
32
33 -- Conteo real
34 SELECT COUNT(*) AS Total_Inner
35 FROM Ventas;
36 GO
37
```
- Query Results:** A table with 5 columns: Cliente, Producto, Fecha, Cantidad, and Total_Inner. It contains 10 rows of data, including entries for '193 feid', '194 SHAKIRA MEBARAK', '195 Nairo Quintana', etc.
- Terminal:** Shows the command 'diplomado > reto_5_6_dgd_2026-main > 02_sql > 2_performance_lab > solution_tuning.sql'.
- Chat:** A sidebar on the right with the heading 'Build with Agent' and a prompt 'Describe what to build next'.

Posteriormente, se ejecutó una consulta utilizando **INNER JOIN**, relacionando correctamente las tablas a través de la tabla **Ventas**.

The screenshot displays a Windows development environment with a code editor and a terminal. The code editor shows a SQL query being executed in a terminal window. The query is as follows:

```
USE LegacyRetail;
GO

SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO

SELECT
    c.Nombre AS cliente,
    p.Nombre AS Producto
FROM Clientes c
JOIN Productos p ON c.ProductoID = p.ProductoID;
```

The terminal output shows the execution of the query, including the SQL Server Execution Times and the results of the query. The output is as follows:

```
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 2 ms.

Started executing query at Line 14
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

(200 rows affected)

Table 'WorkFile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0.
Table 'Products'. Scan count 1, logical reads 4, physical reads 0, page server reads 0, read-ahead reads 0.
Table 'Clientes'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead reads 0.

SQL Server Execution Times:
CPU time = 20 ms, elapsed time = 20 ms.

Started executing query at Line 26
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.

(1 row affected)

Table 'Products'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead reads 0.
Table 'Clientes'. Scan count 1, logical reads 2, physical reads 0, page server reads 0, read-ahead reads 0.
```

The interface includes a sidebar with a file explorer showing the project structure, a search bar, and a terminal window. The terminal window is titled "Messages" and shows the execution of the query. The code editor is titled "solution_tuning.sql" and shows the SQL query being executed.

8 / 10

4. Comparación de Resultados

Aspecto	CROSS JOIN	INNER JOIN
Relación de negocio	No existe	Correcta
Filas generadas	Excesivas	Necesarias
Lecturas lógicas	Altas	Reducidas
Rendimiento	Bajo	Optimizado

5. Conclusiones y Aprendizajes

- La normalización permitió organizar la información de manera clara y estructurada.
- Separar las entidades redujo la redundancia y mejoró la calidad de los datos.
- Un mal uso de CROSS JOIN puede generar graves problemas de rendimiento.
- El uso adecuado de INNER JOIN mejora considerablemente la eficiencia de las consultas.
- El análisis de métricas es fundamental para justificar decisiones técnicas.

Este proyecto permitió comprender la importancia del diseño de bases de datos y cómo este impacta directamente en el rendimiento y la estabilidad de un sistema.

Diagrama Entidad–Relación (DER)

El siguiente diagrama representa el modelo relacional normalizado a Tercera Forma Normal (3FN), donde las entidades Clientes, Productos y Sucursales se relacionan a través de la tabla Ventas, la cual actúa como tabla central del modelo.

