**1. HOOK: EL RETO DE BRICKVISTA Y POR QUÉ MICROSOFT FABRIC**

BrickVista S.A. es una empresa dedicada al desarrollo y comercialización de proyectos inmobiliarios residenciales y corporativos en Latinoamérica. Con una presencia creciente en el mercado, enfrentaba desafíos críticos en la gestión y análisis de sus datos:

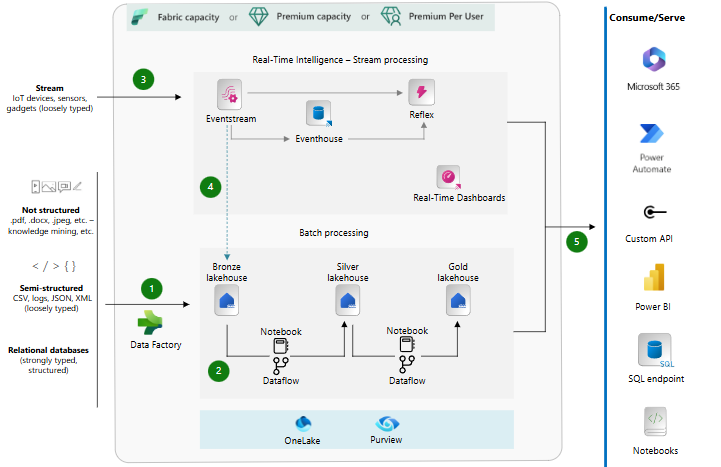
* Sus fuentes estaban distribuidas entre SQL Server y archivos planos, lo que fragmentaba la información.
* Los reportes de negocio se elaboraban manualmente en Excel, consumiendo hasta 48 horas de trabajo.
* No existía una vista unificada del inventario inmobiliario, lo que afectaba la toma de decisiones sobre precios, campañas, propiedades y brokers.

Para resolver estos problemas, BrickVista optó por una transformación completa hacia una plataforma data-driven utilizando Microsoft Fabric. ¿Por qué Fabric?

* **Unificación:** integra ingesta, almacenamiento, procesamiento, análisis y visualización en un solo entorno.
* **Escalabilidad:** permite manejar grandes volúmenes de datos con modelos robustos y notebooks PySpark.
* **Visualización** **integrada** con Power BI, facilitando dashboards interactivos y análisis en tiempo real.
* **Gobernanza y seguridad**: gestión de datos personales auditables, cumpliendo normativas de privacidad.

Fabric se convierte así en el eje central de la estrategia de analítica avanzada de BrickVista, permitiéndoles transformar sus operaciones en decisiones basadas en datos.

**2. ARQUITECTURA (CON DIAGRAMA LÓGICO)**



A continuación, se describe la arquitectura implementada:

**🔄 Ingesta**

* Pipelines en Data Factory (Fabric) conectados a Azure Data Lake Storage Gen2 (ADLS).
* Archivos CSV de diversas entidades (Projects, Sales, Clients, etc.) cargados a Lakehouse / Raw.

**🏠 Lakehouse y Arquitectura Medallion**

* Organización en capas Bronze ➝ Silver ➝ Gold usando notebooks en PySpark.
* Datos transformados y enriquecidos para análisis confiables.

**🧠 Modelo Semántico**

Se construyó un modelo estrella en Fabric Warehouse, con:

* Tablas de hechos: sales, leads.
* Tablas dimensión: clients, projects, brokers, campaigns, properties.

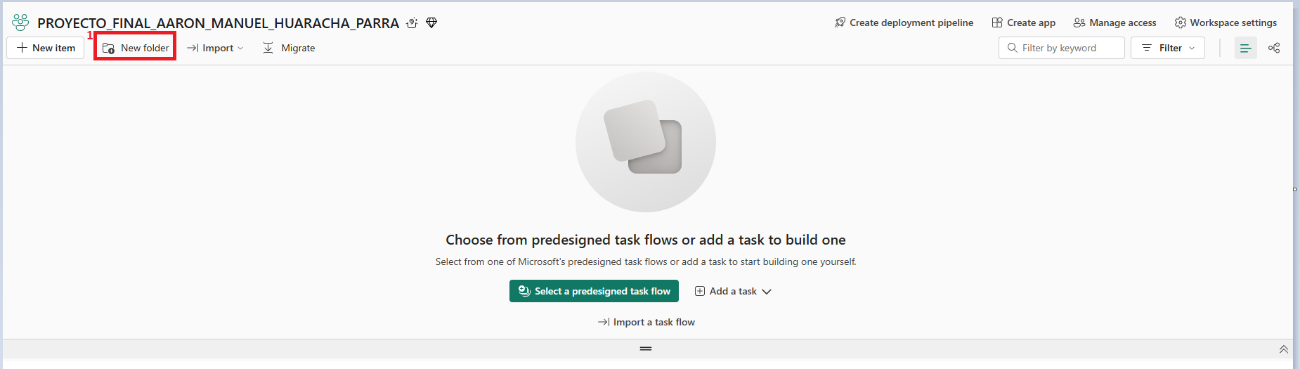
**📊 Visualización**

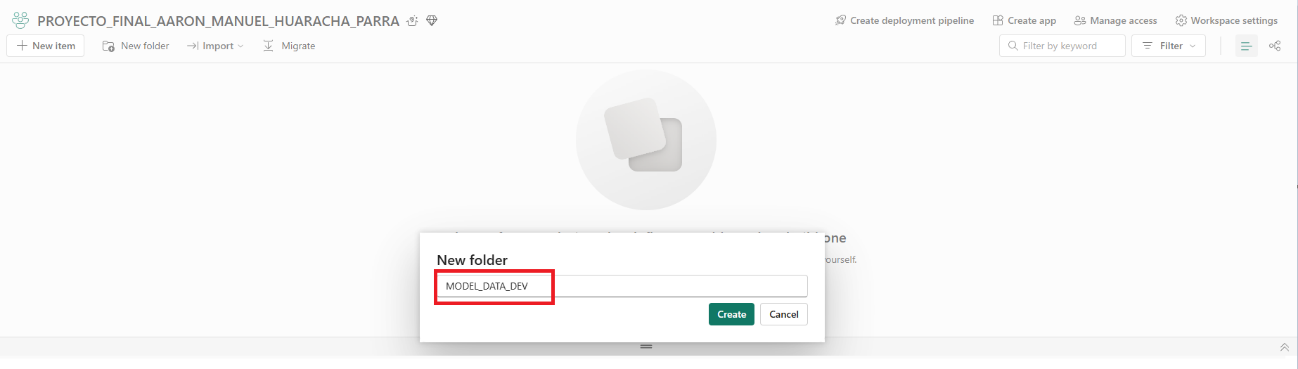
Power BI App: Real Estate Command Center

* Dashboards para análisis de precios, campañas, inventario y clientes.

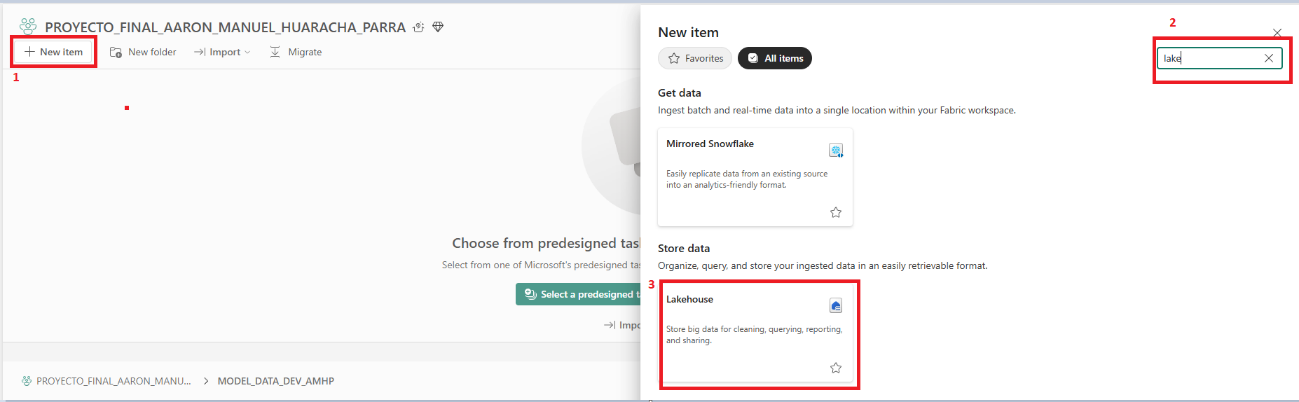
**3. PASO A PASO DEL DESARROLLO, ACOMPAÑADAS DE IMAGENES**

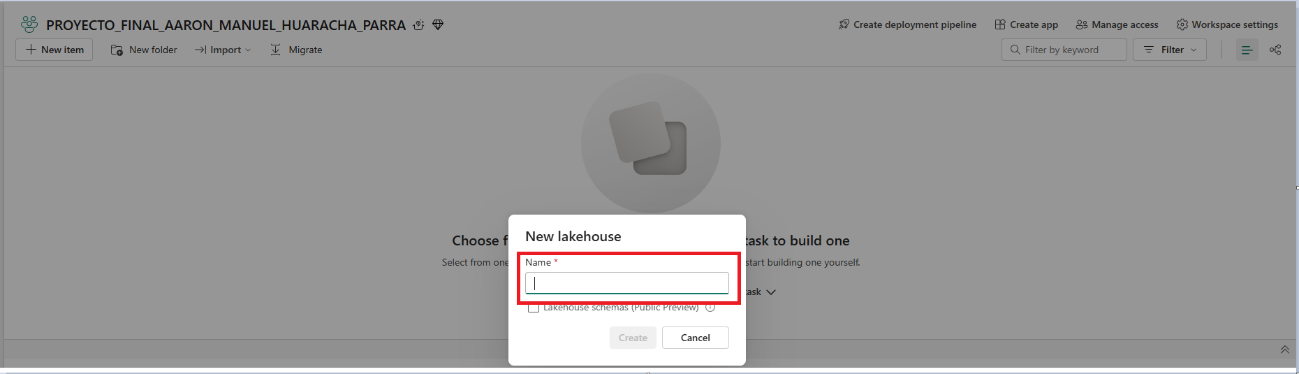
**ARQUITECTURA**

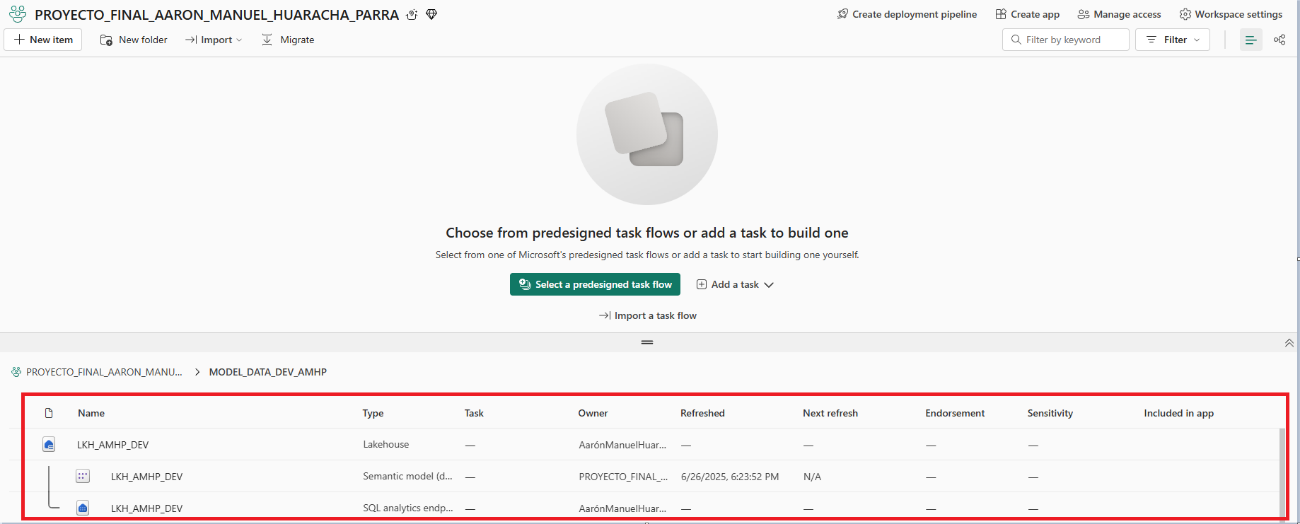
Creamos un folder para poder almacenar nuestros ítems que vamos a utilizar para este proyecto. Le daremos click donde dice “New Folder”.

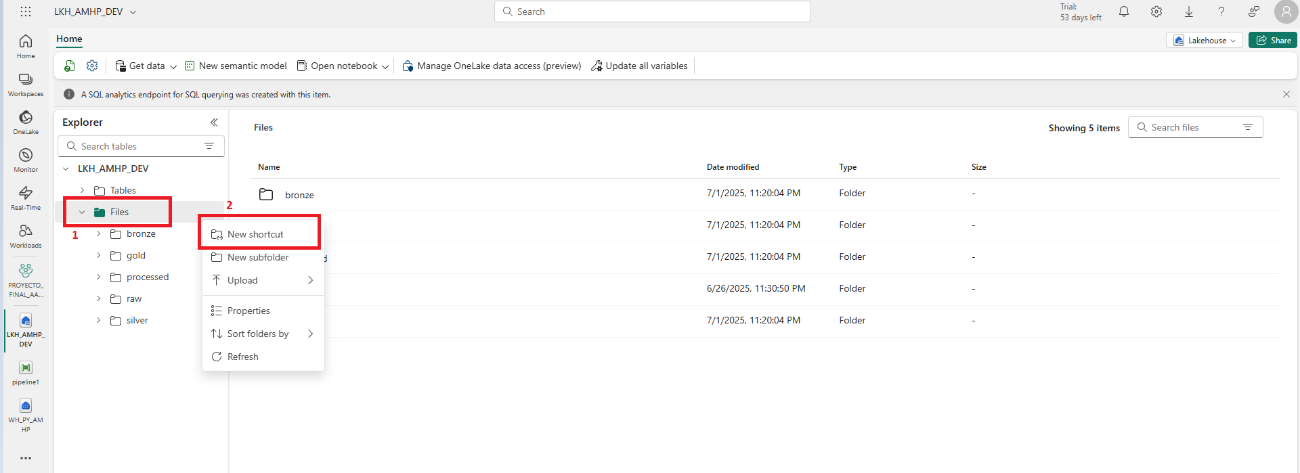
Nos saldrá esta ventana donde vamos a colocar el nombre de nuestro proyecto.

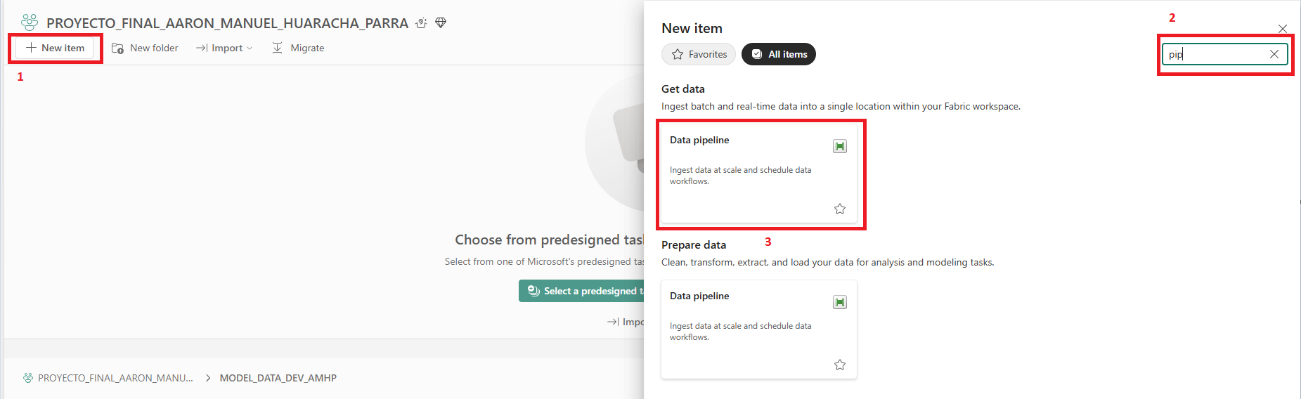
Ingresamos a nuestro folder creado donde crearemos nuestro lakehouse. Este lakehouse será la base para las transformaciones posteriores en las capas Bronze, Silver y Gold.

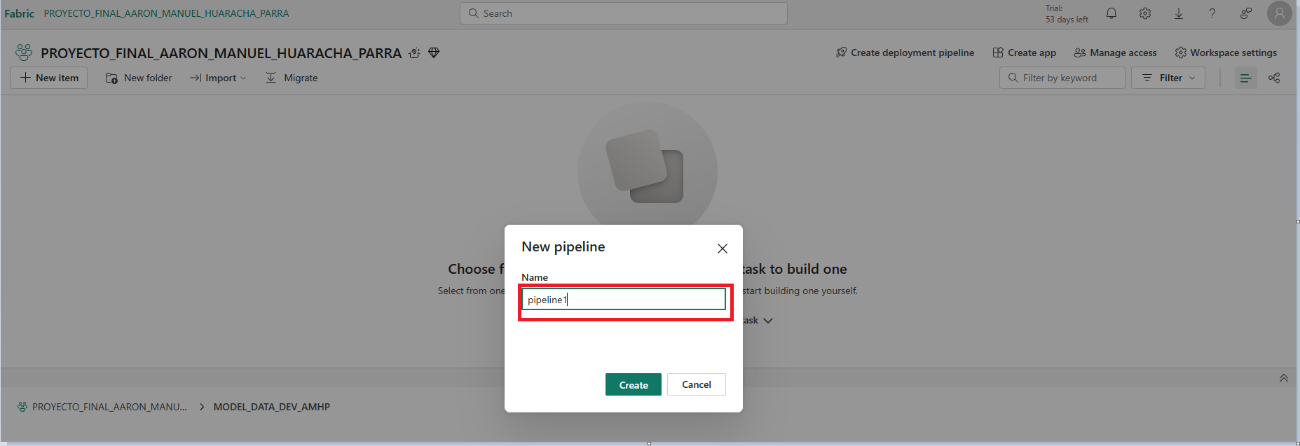


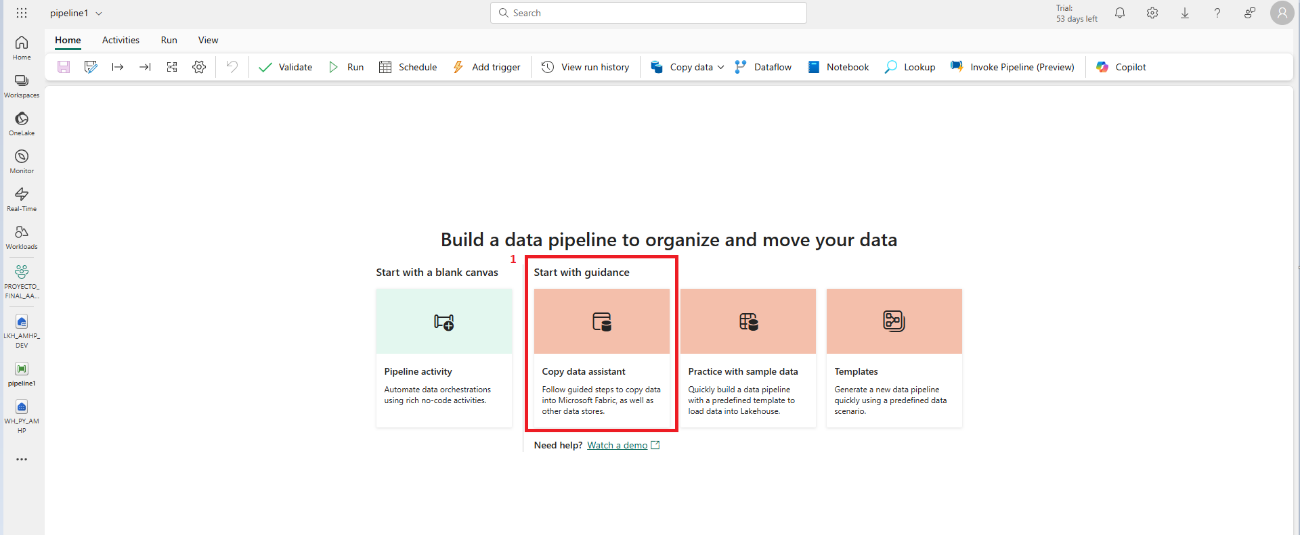
Nos saldrá una ventana donde le pondremos el nombre a nuestro lakehouse.

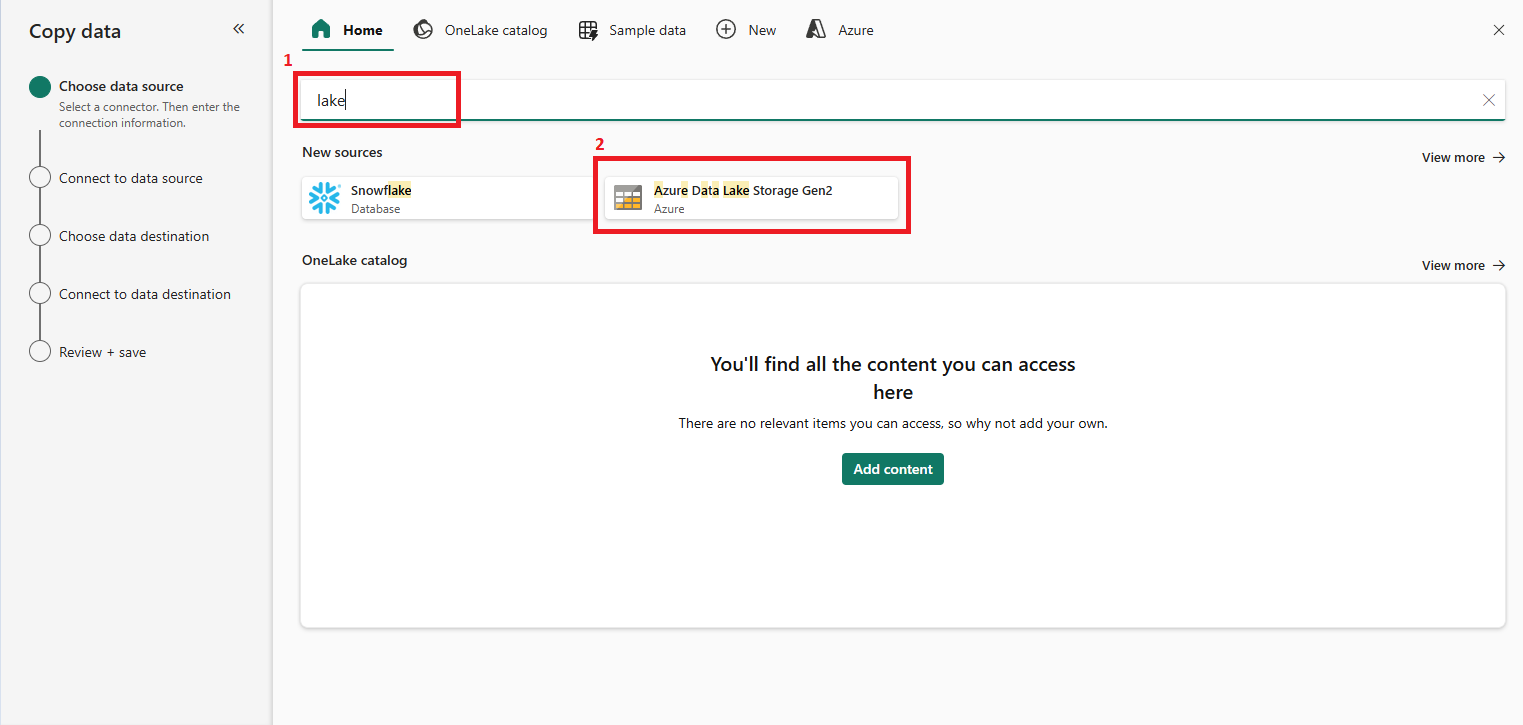
Cuando le damos en “Create” nos saldrá esta estructura de nuestro lakehouse.

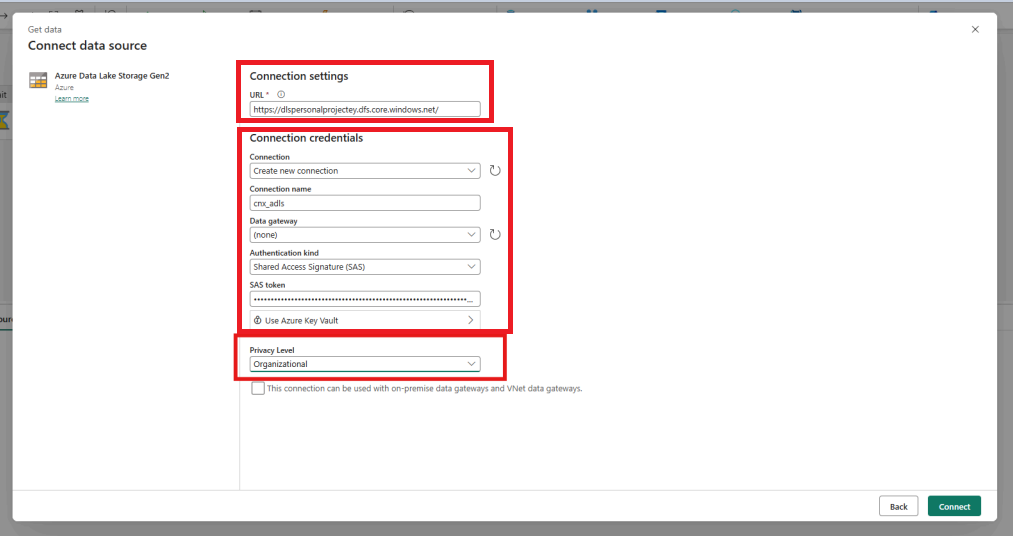
Le damos click a nuestro lakehouse para poder crear una subcarpeta llamada “Raw” donde vamos almacenar nuestros archivos CSV que nos descargaremos.

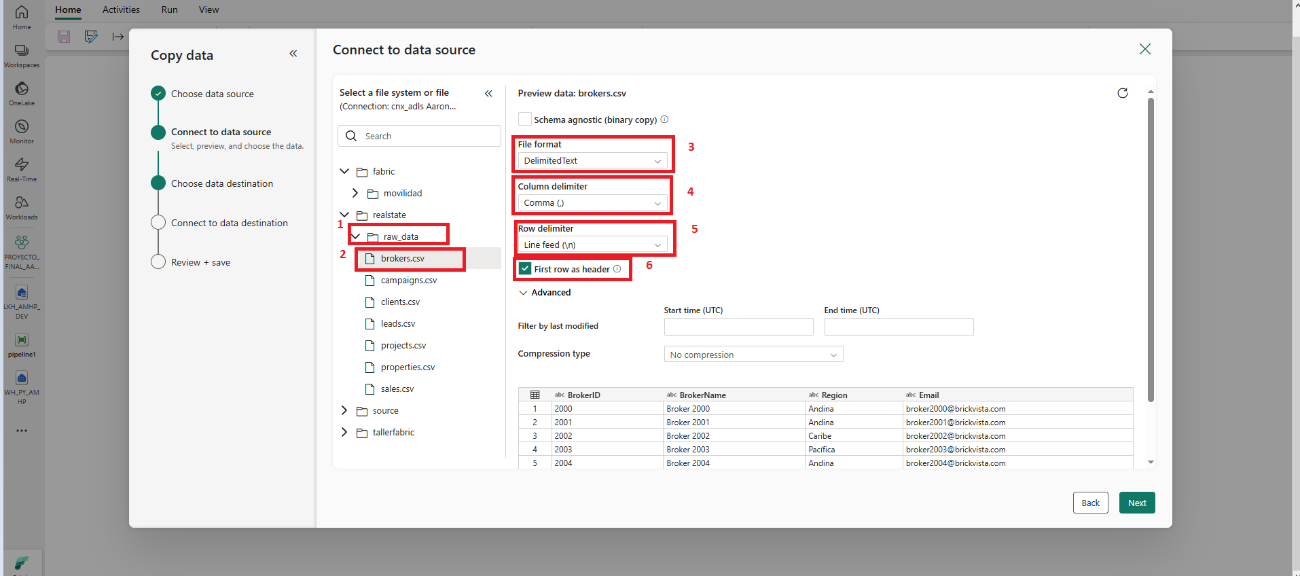
Ahora crearemos nuestro pipeline se encargará de extraer los datos desde la fuente original ADLS y cargarlos al Lakehouse previamente creado en nuestra carpeta Raw.

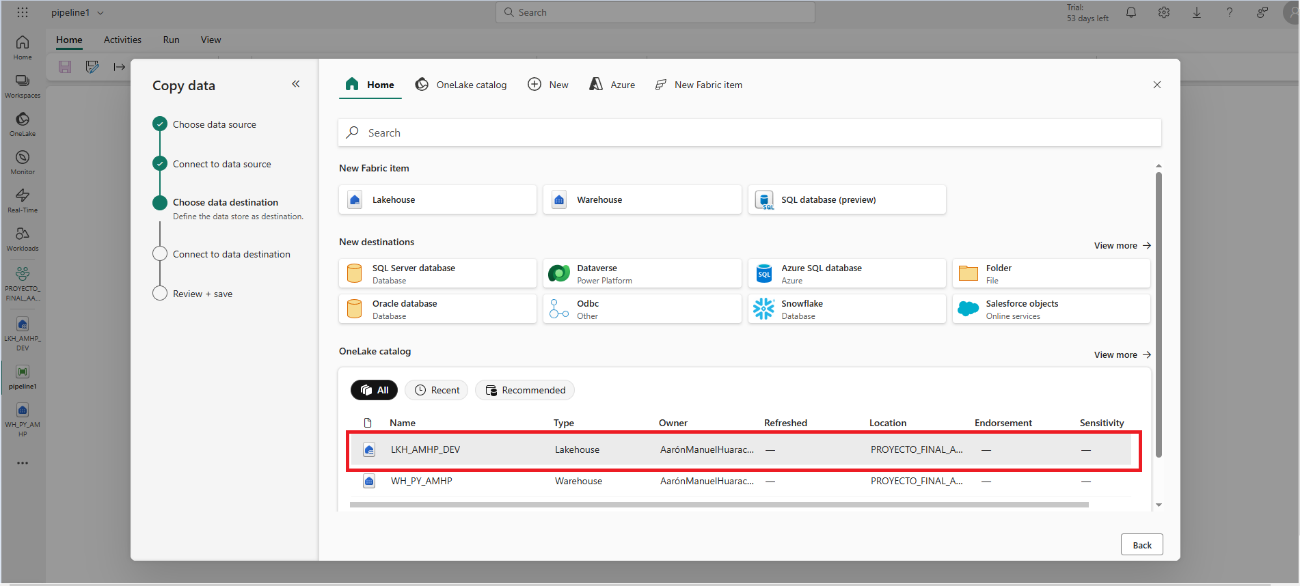
Le colocamos un nombre a nuestro pipeline.

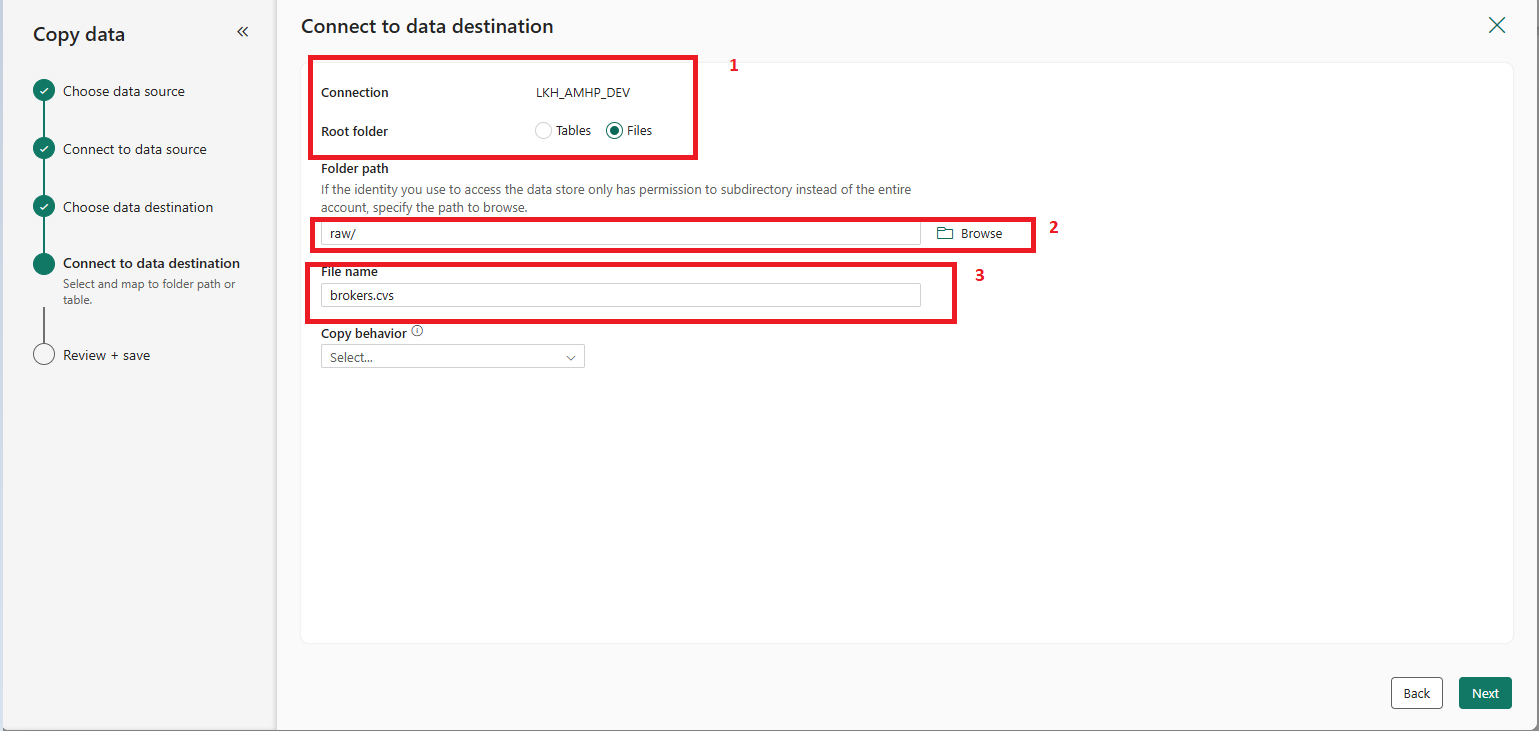
Nos saldrá esta interfaz donde vamos hacer click en “Copy data assistant”

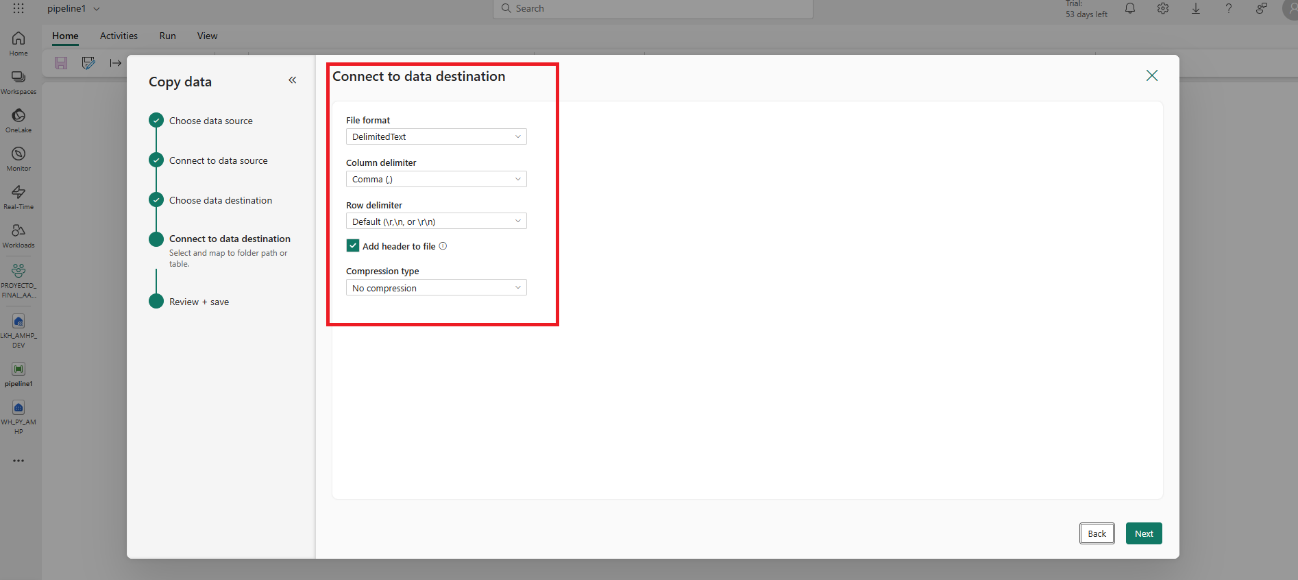
Buscamos el ítem Azure Data Lake Storage Gen2 para poder obtener los datos que vamos a colocarlos en nuestros Lakehouse.

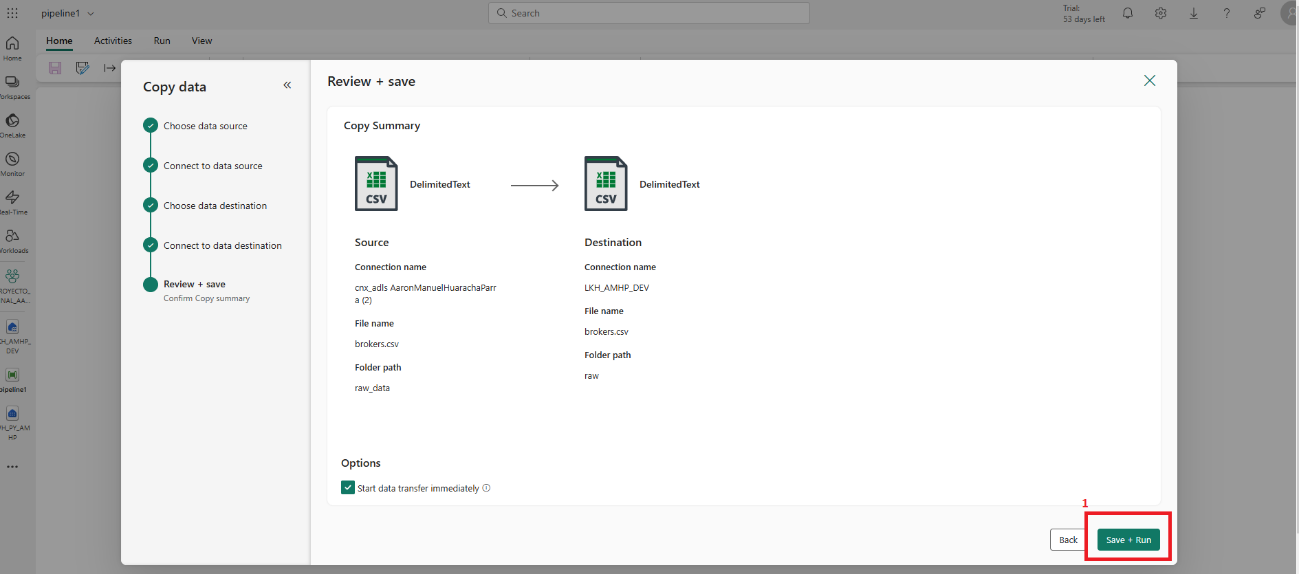
Nos saldré este recuadro donde vamos a colocar nuestras credenciales para poder extraer los datos.

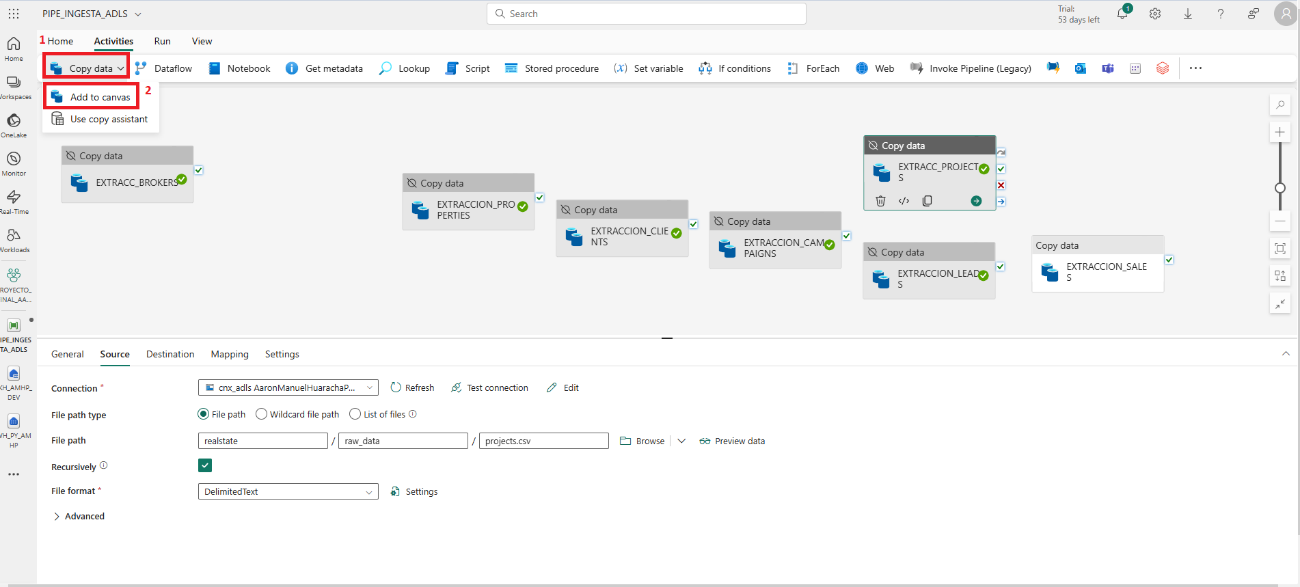
Seleccionamos las carpetas donde vamos a extraer los archivos CSV.

Seleccionamos nuestro Lakehouse para poder almacenar los archivos CSV.

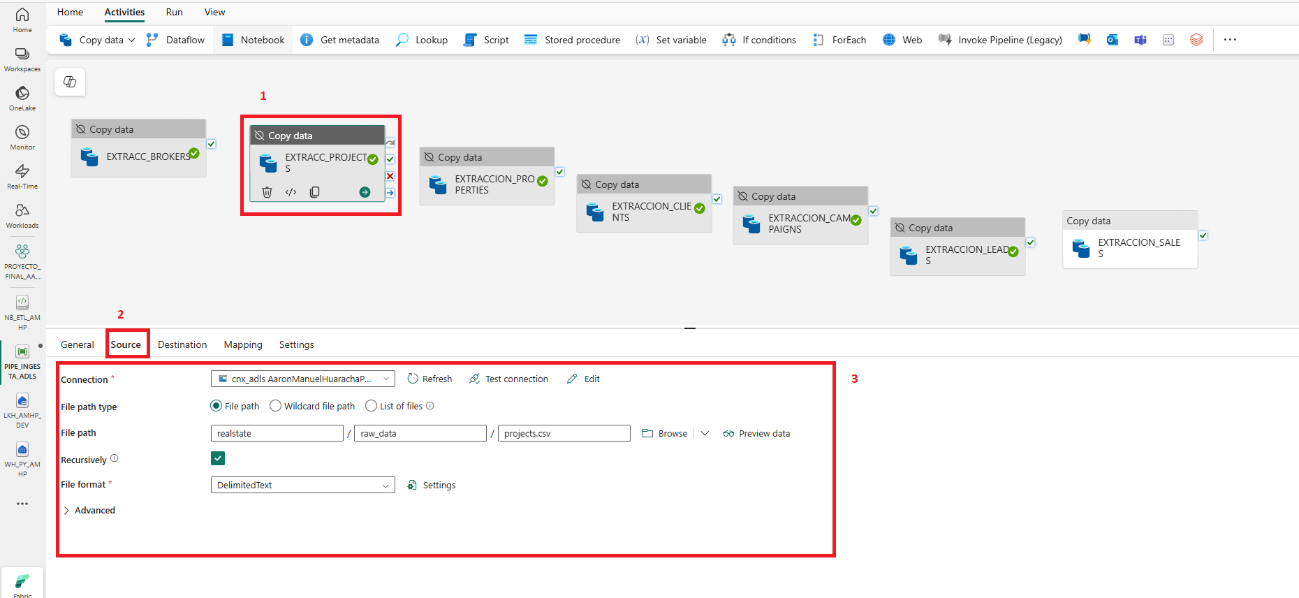
Aquí le colocamos la ruta donde vamos a querer que se almacene nuestro CSV.

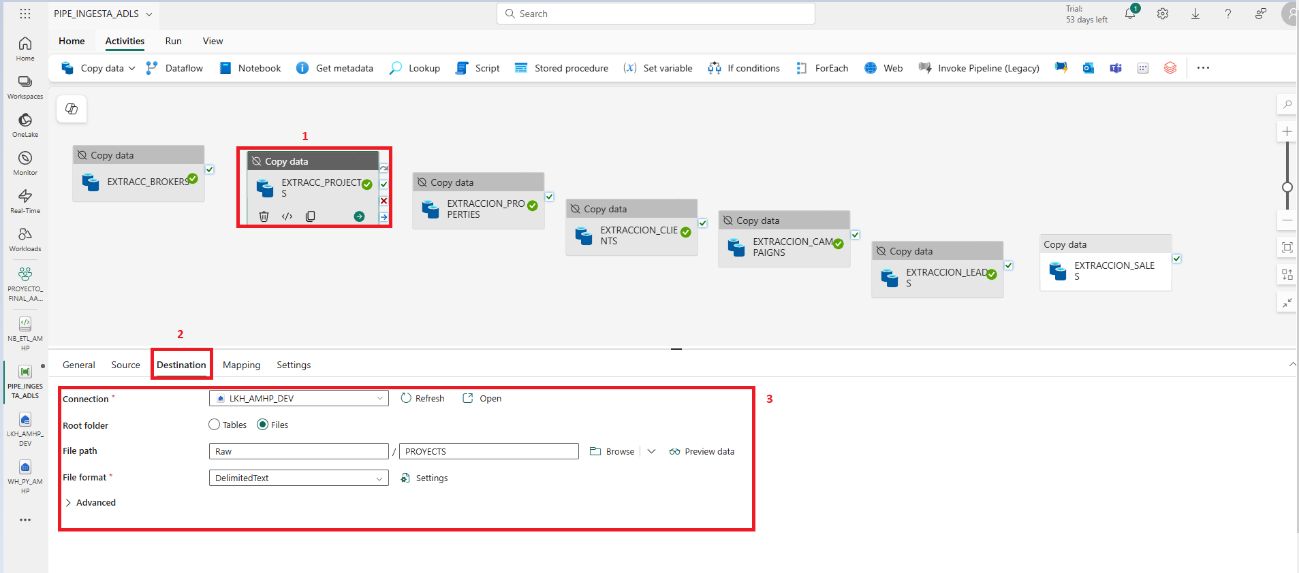
Dejamos con esta configuración previa.

Le damos click en “Save + Run”.

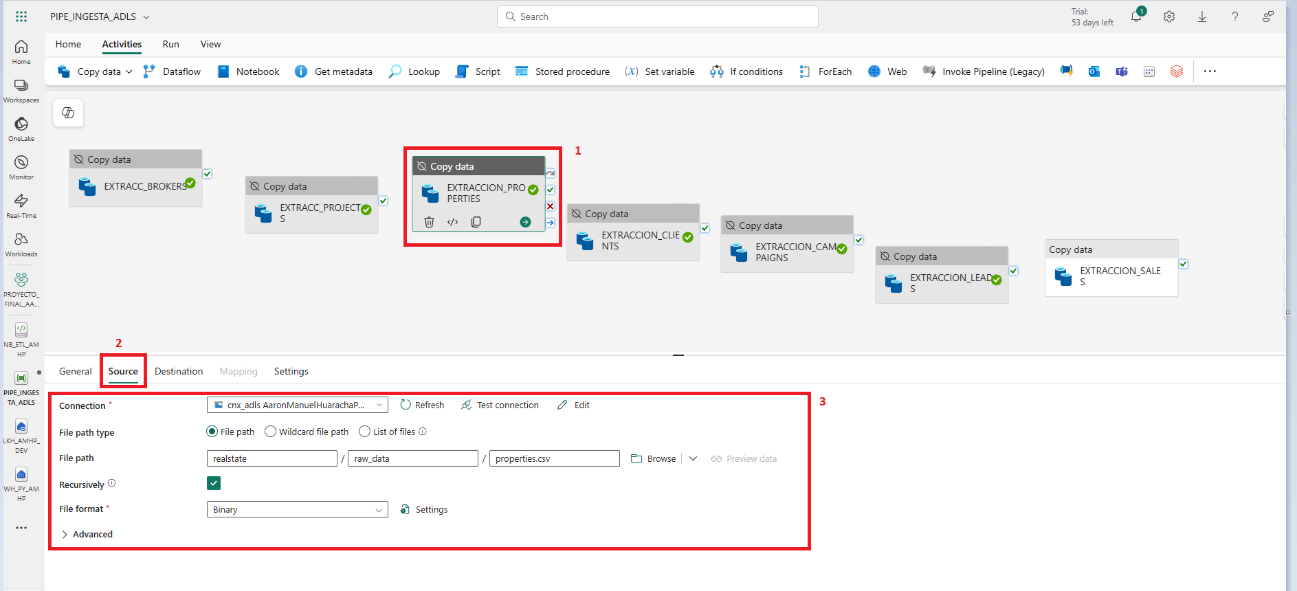
Creamos nuevas actividades de tipo 'Copy Data' para los archivos CSV restantes, extrayéndolos directamente desde el contenedor de ADLS hacia el Lakehouse.

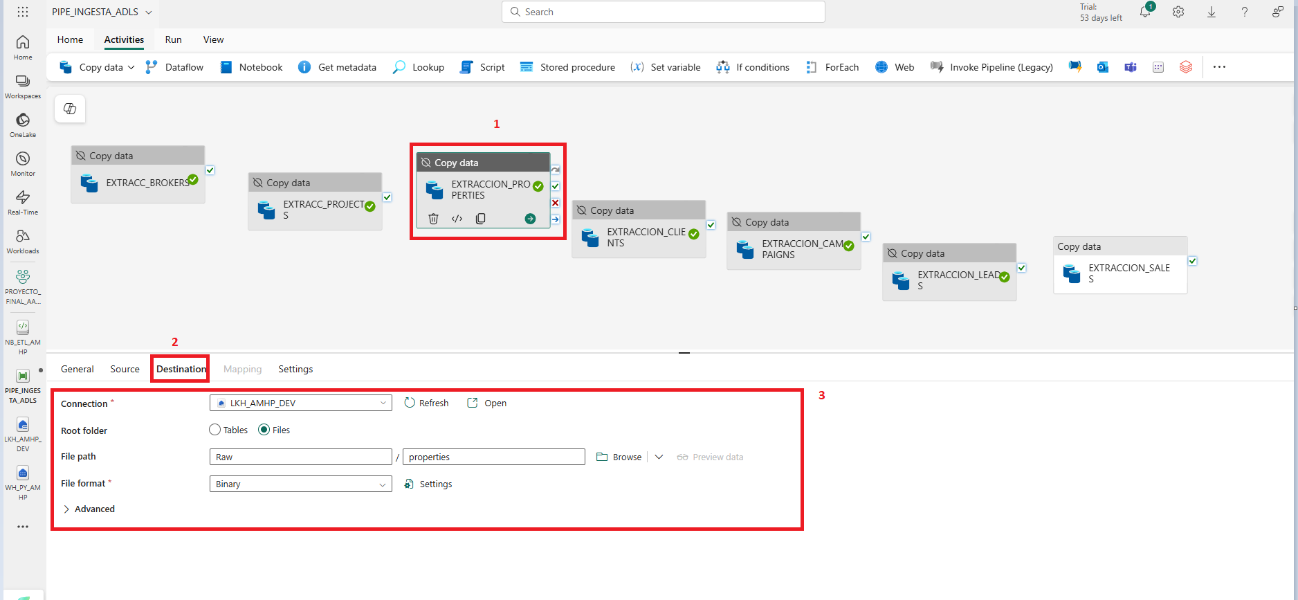
Colocamos a estas configuraciones para cada archivo CVS.

Projects:

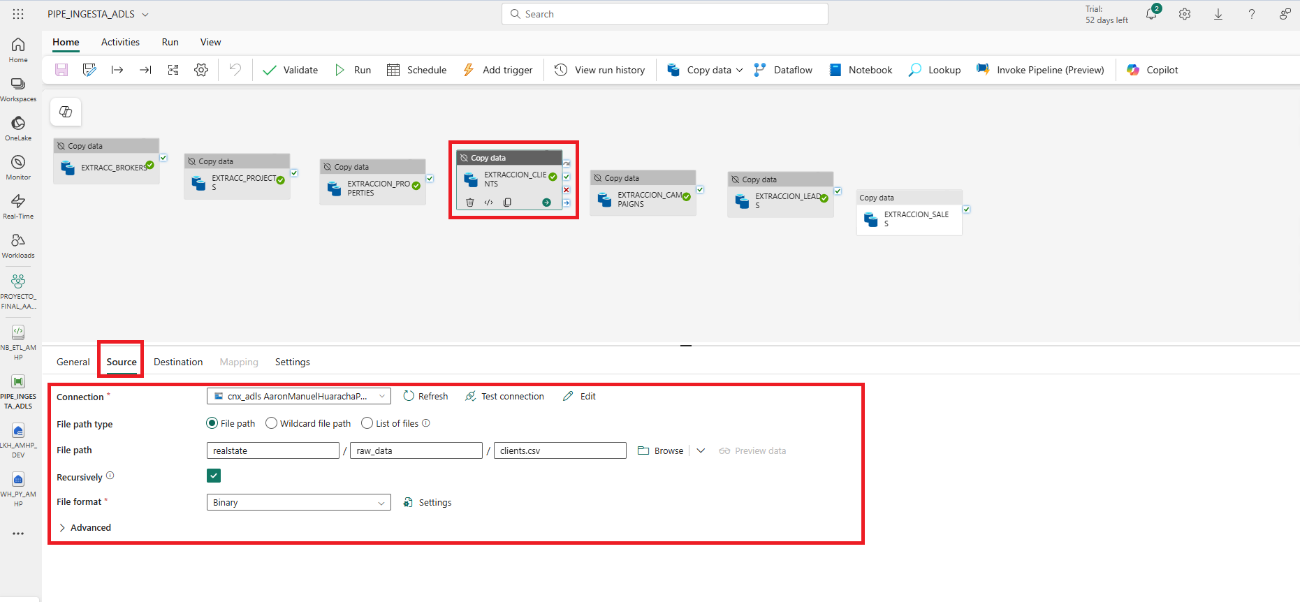


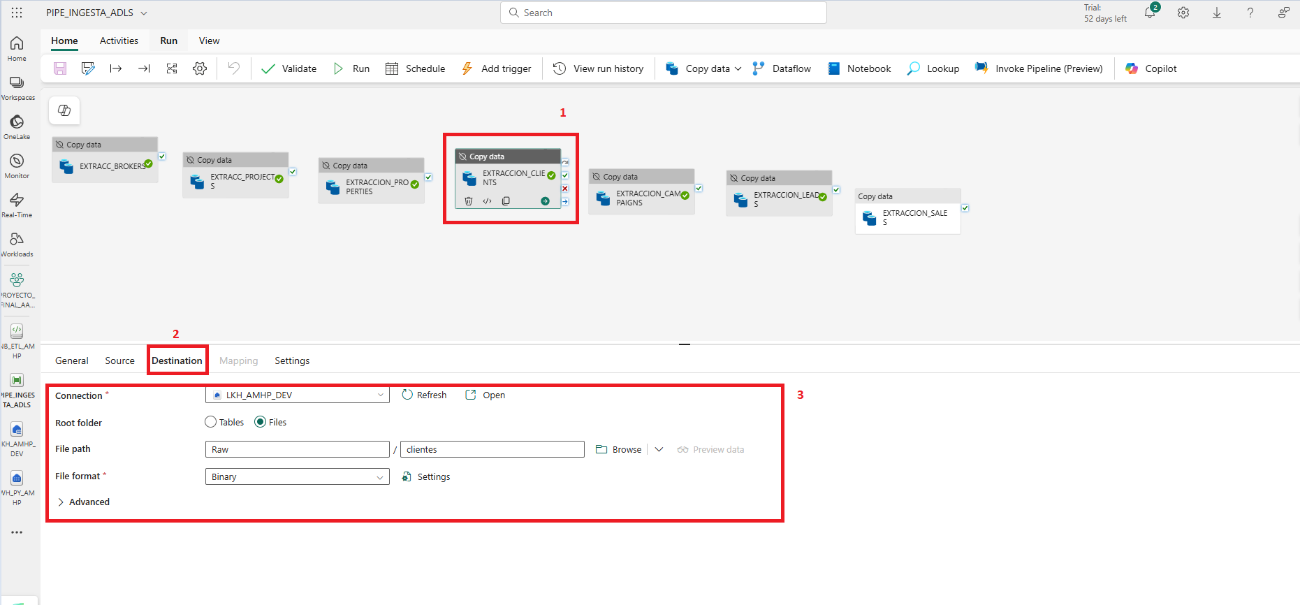
Properties:



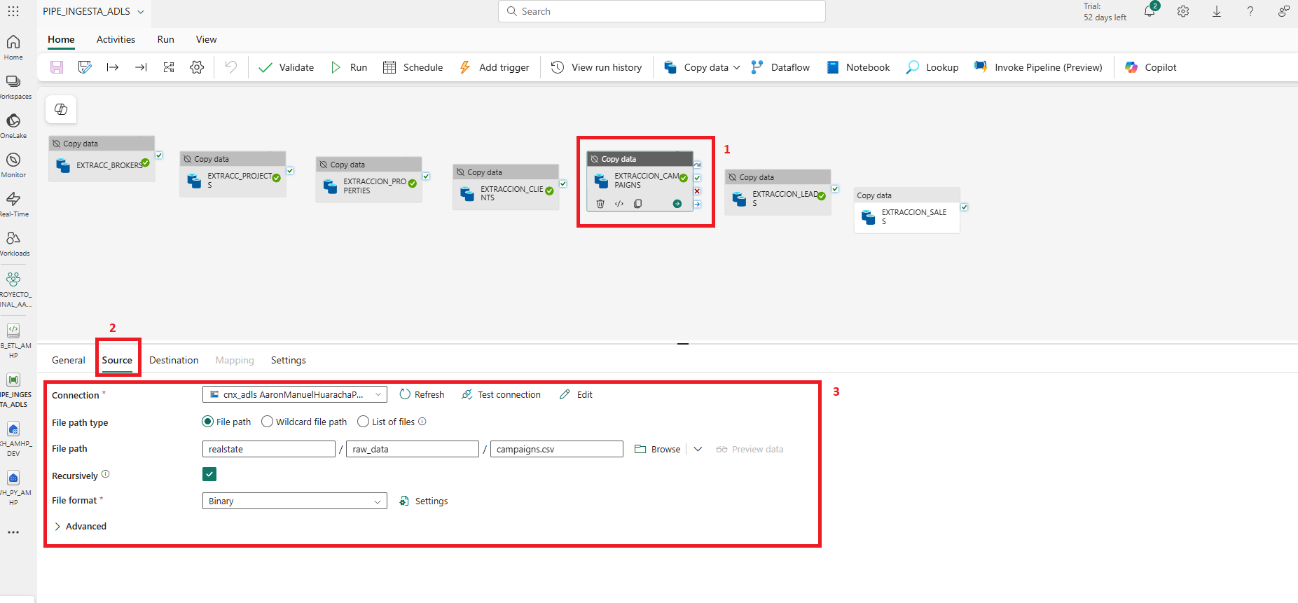


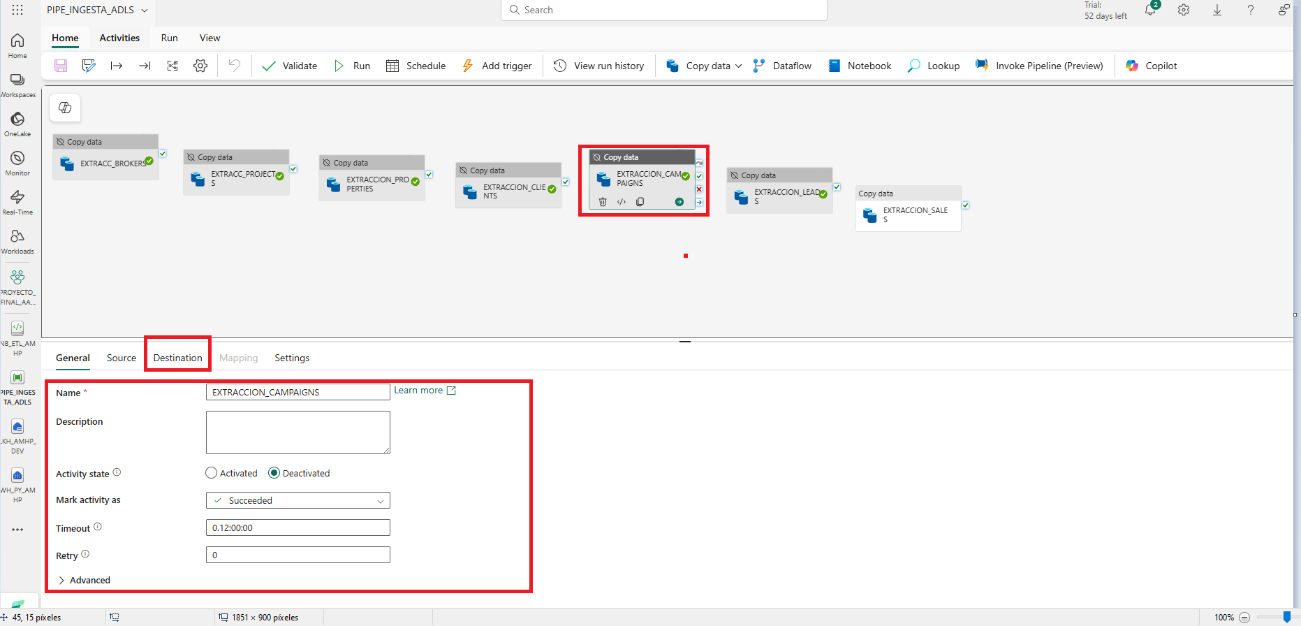
Clients:



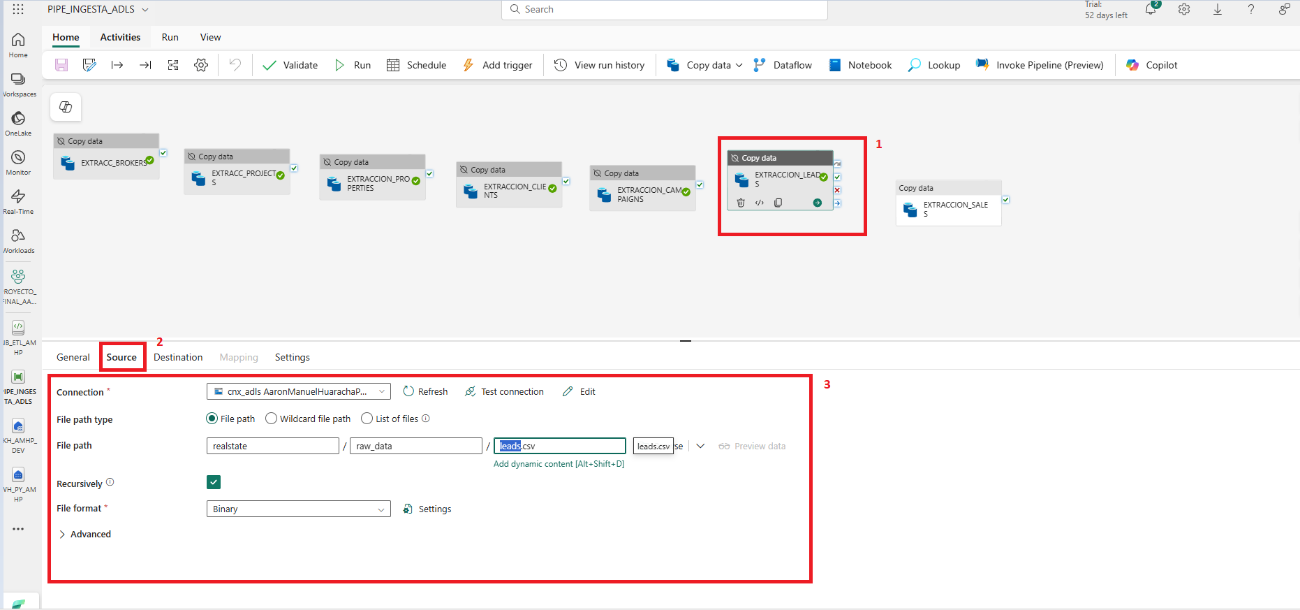


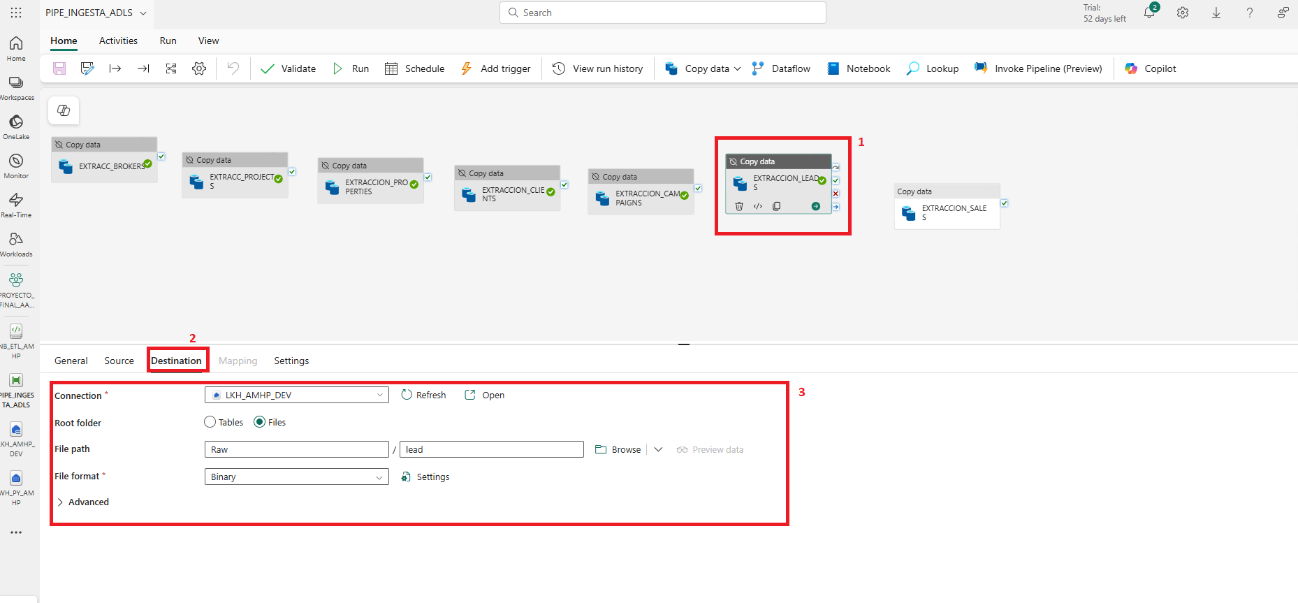
Campaigns:



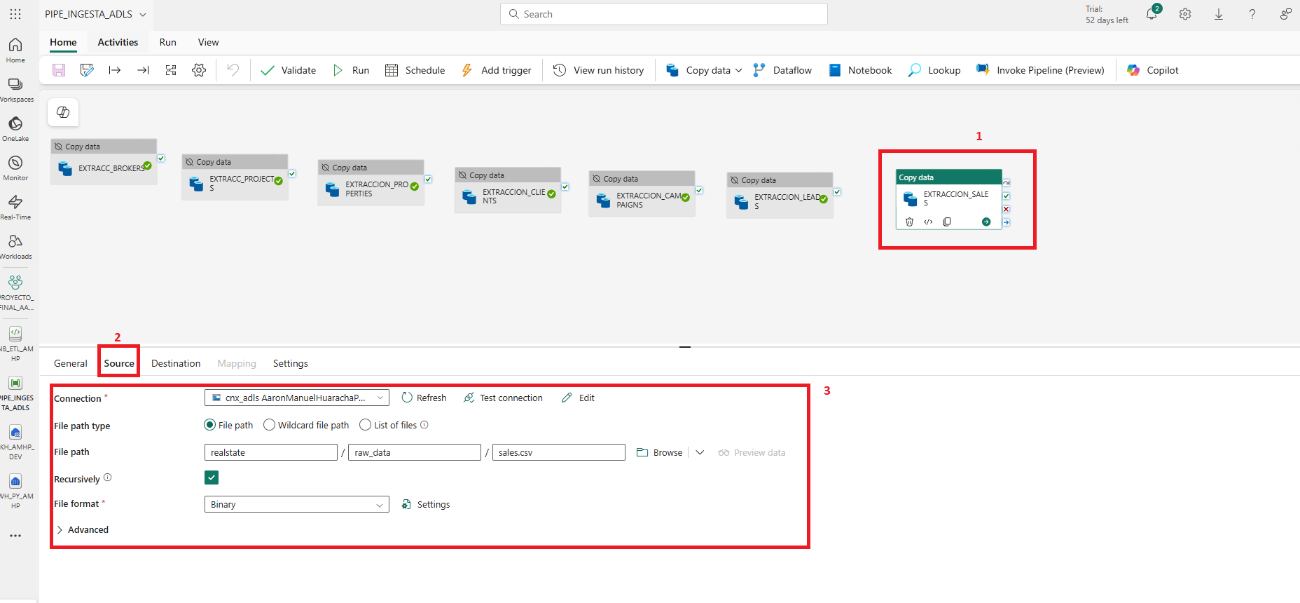


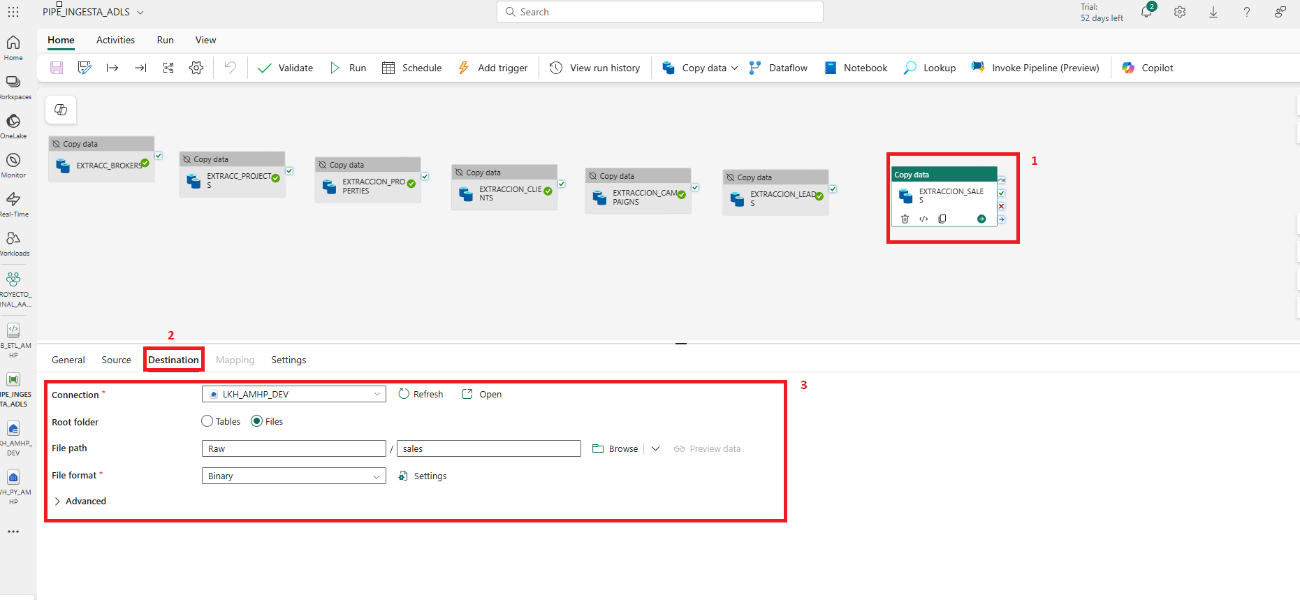
Leads:

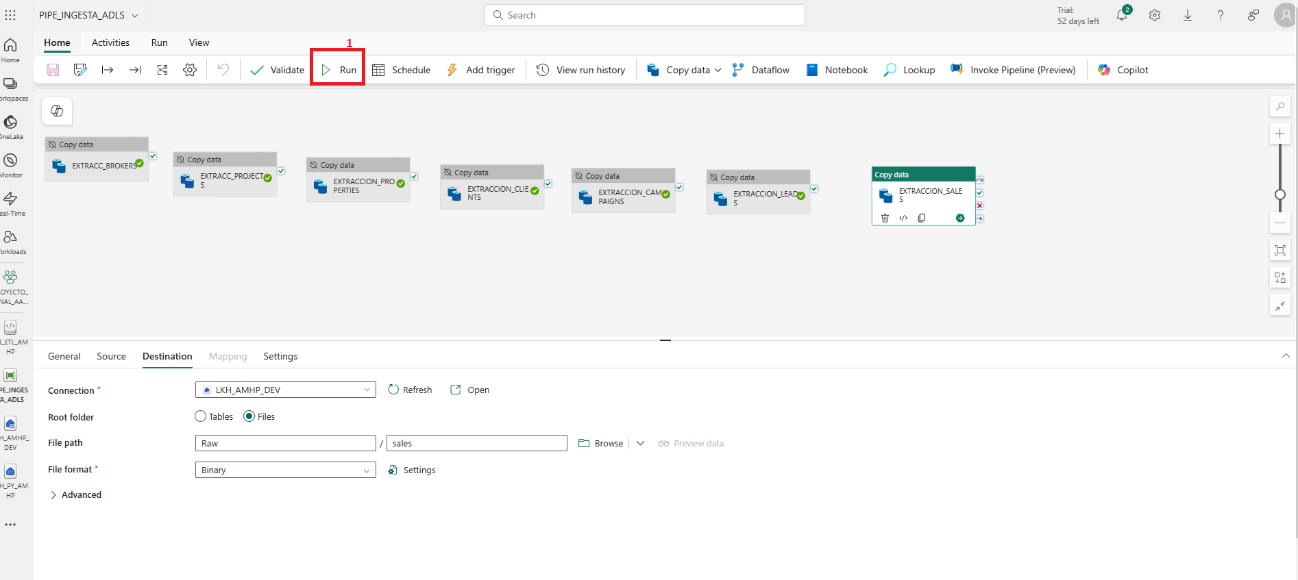




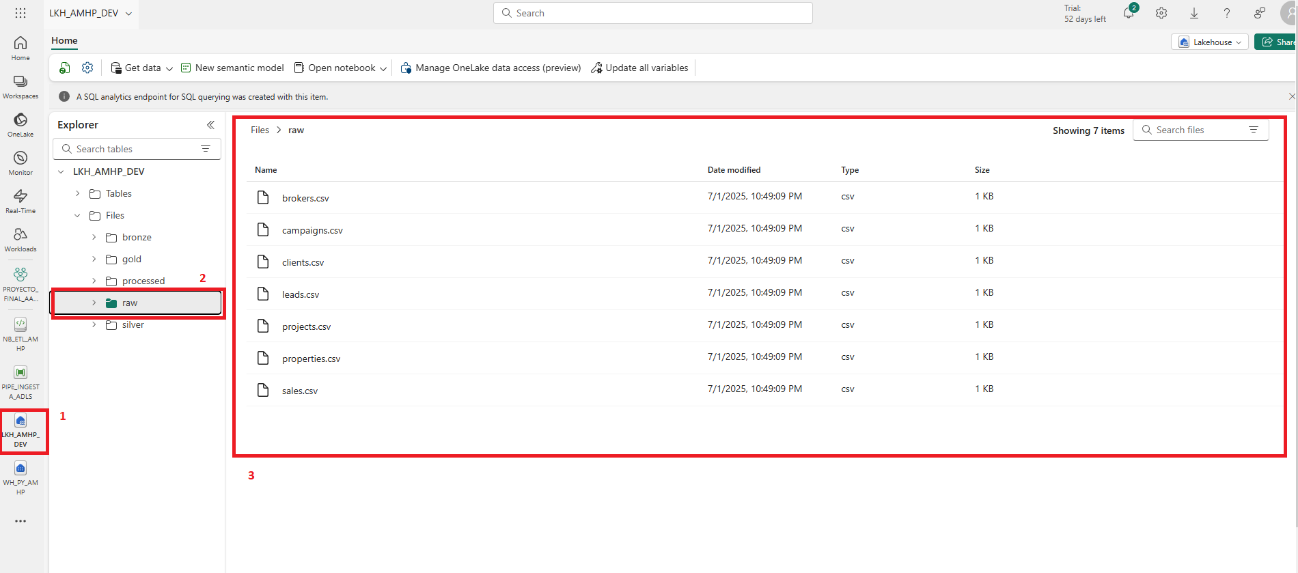
Sales:

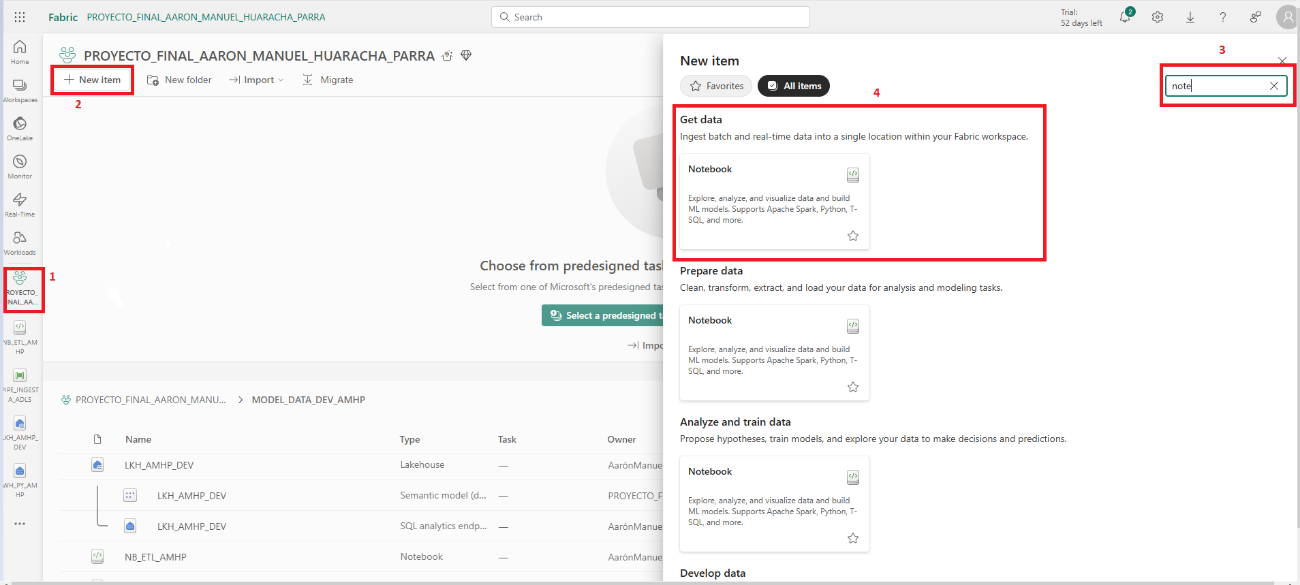




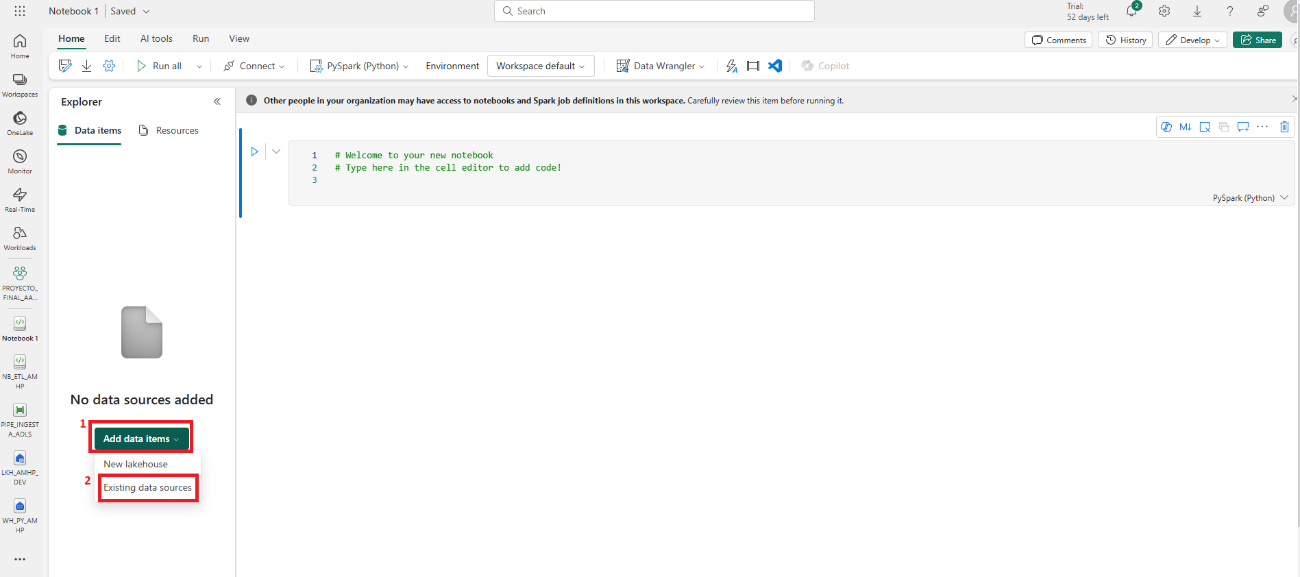
Ejecutamos nuestro pipeline.

Ahora, si accedemos a nuestro Lakehouse, podemos visualizar que los archivos CSV se han almacenado correctamente en la carpeta 'Raw'.

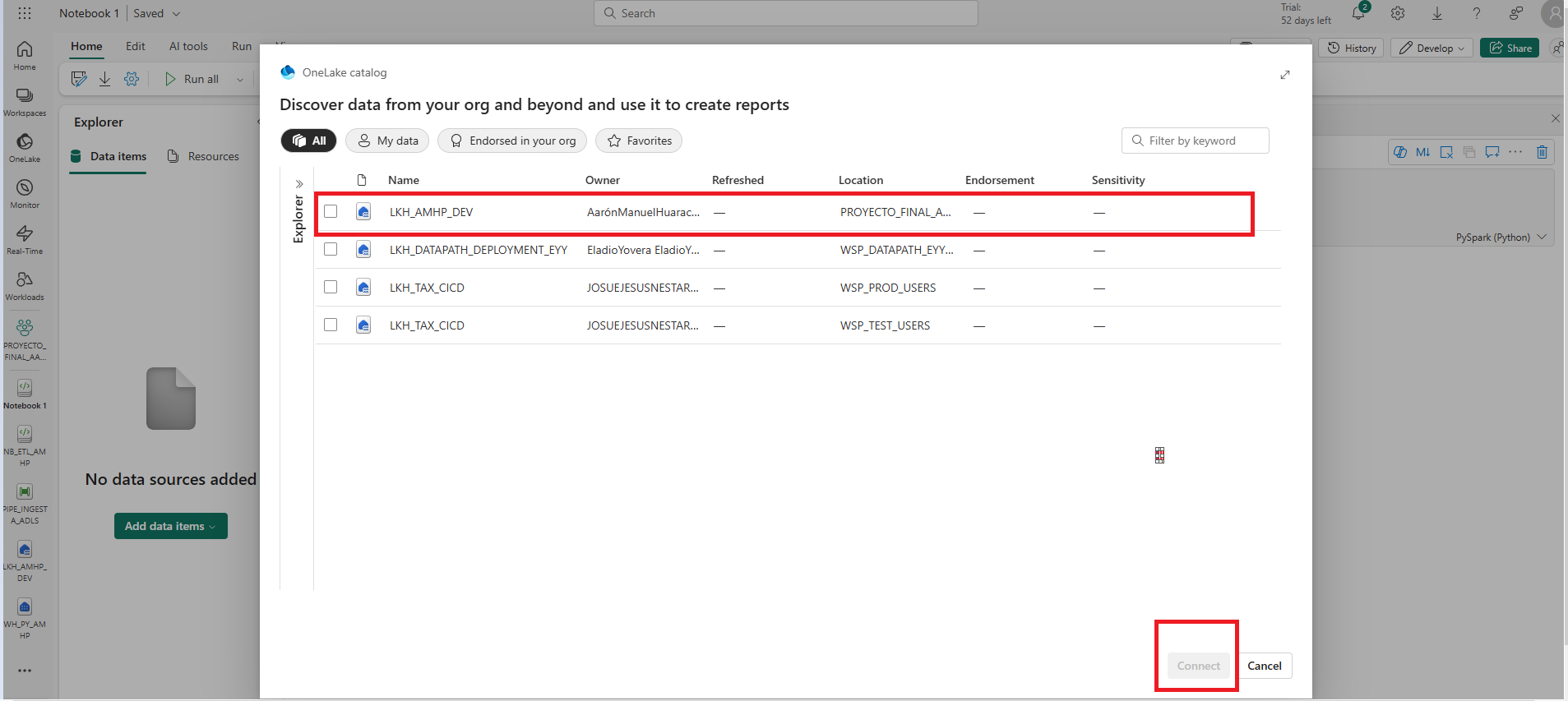


A continuación, retornamos a la carpeta principal del proyecto para crear un notebook que nos permitirá realizar la limpieza de datos y la creación de las tablas necesarias a partir de los archivos CSV previamente cargados.

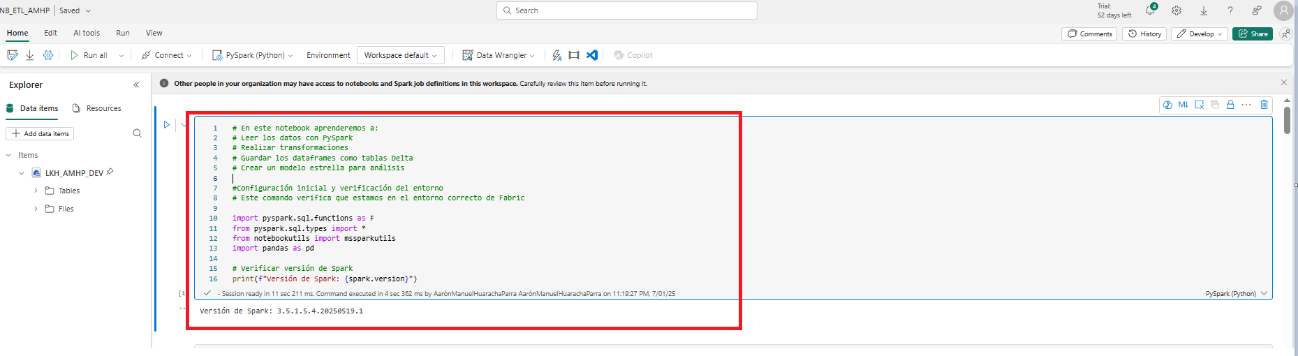
Aparecerá una interfaz desde la cual podremos seleccionar el Lakehouse que contiene los archivos CSV previamente cargados durante la fase de ingesta.

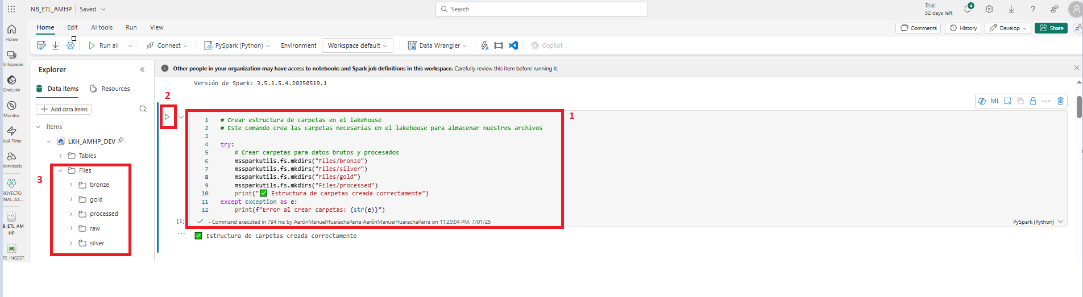


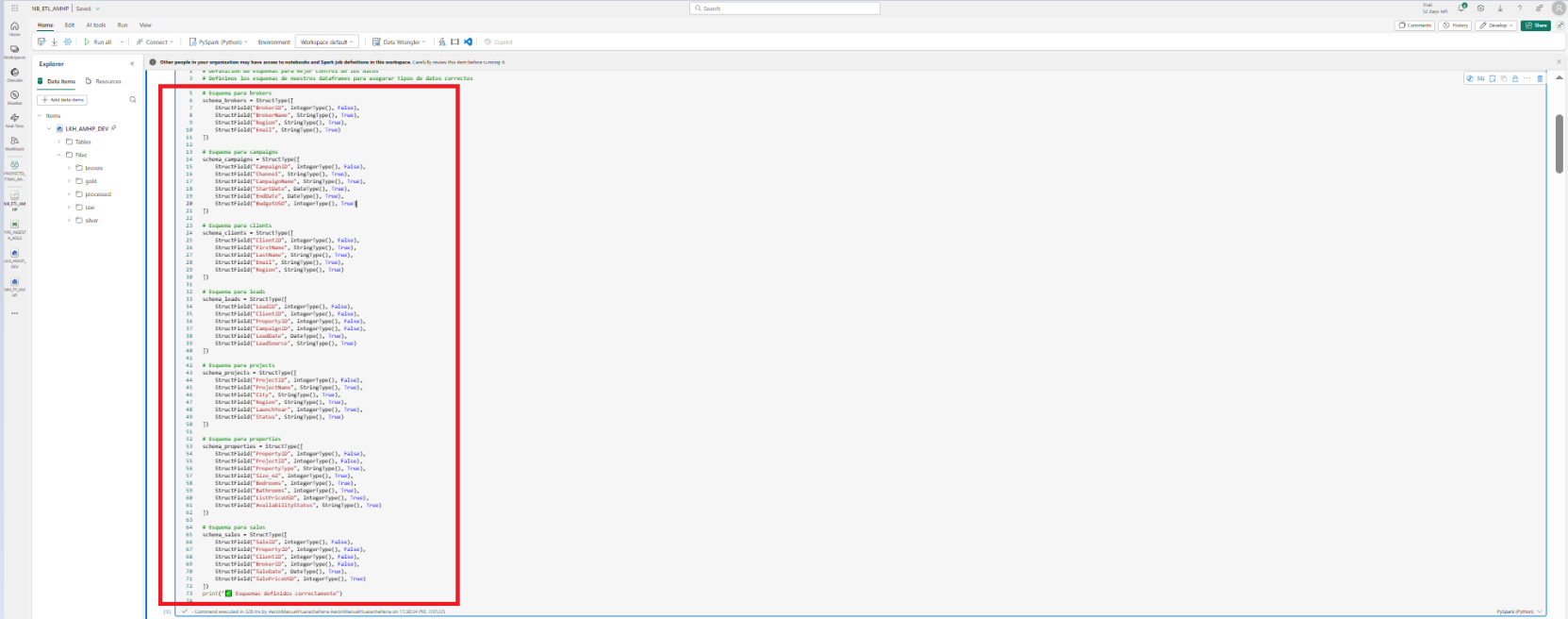
Seleccionamos nuestro Lakehouse

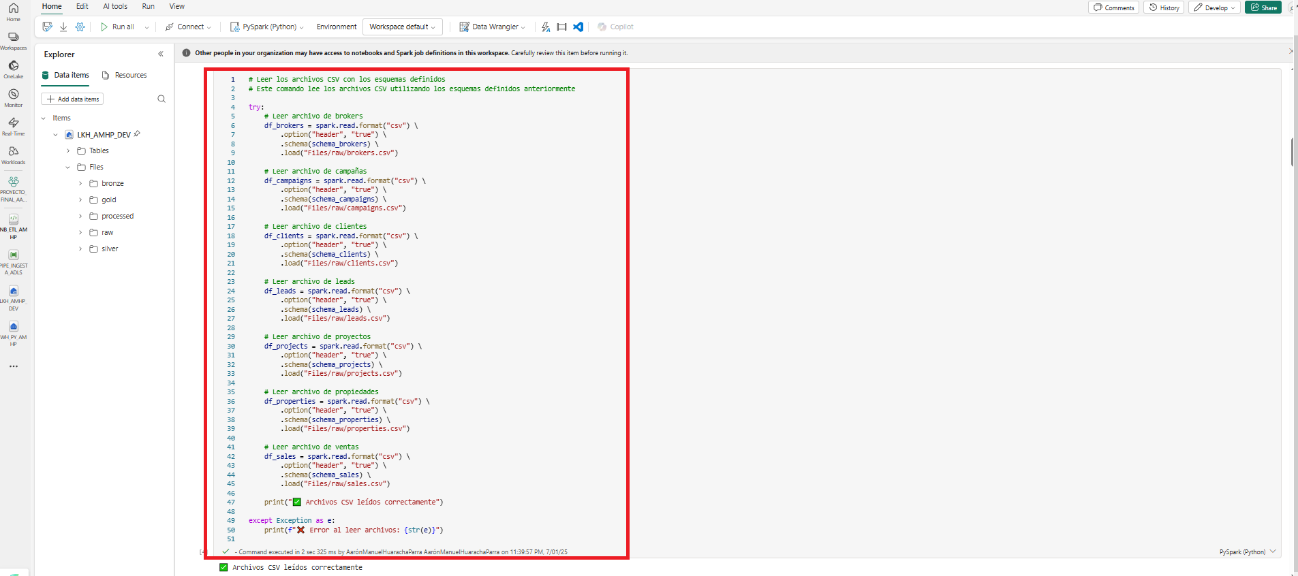


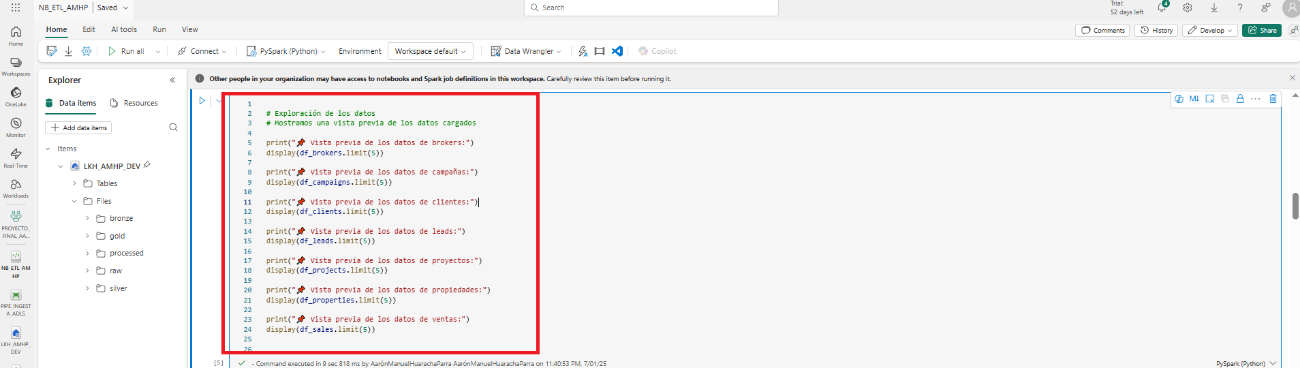
A continuación, desarrollaremos código en PySpark orientado a la limpieza, extracción y transformación de los datos, con el objetivo de generar las tablas estructuradas requeridas para el proyecto.

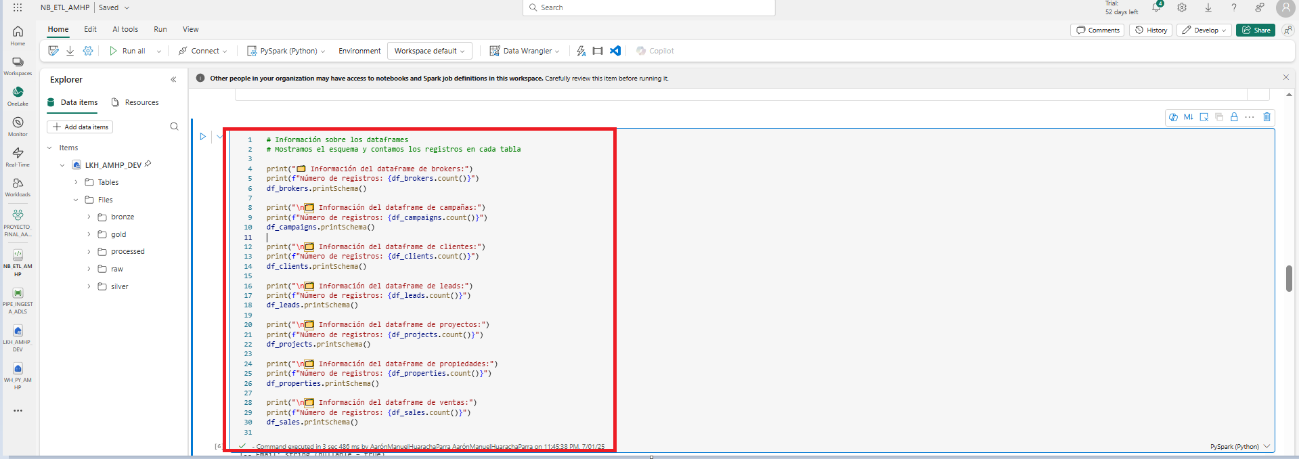
Este bloque configura el entorno de trabajo en Microsoft Fabric y asegura que Spark esté funcionando correctamente antes de leer y transformar los datos con PySpark.

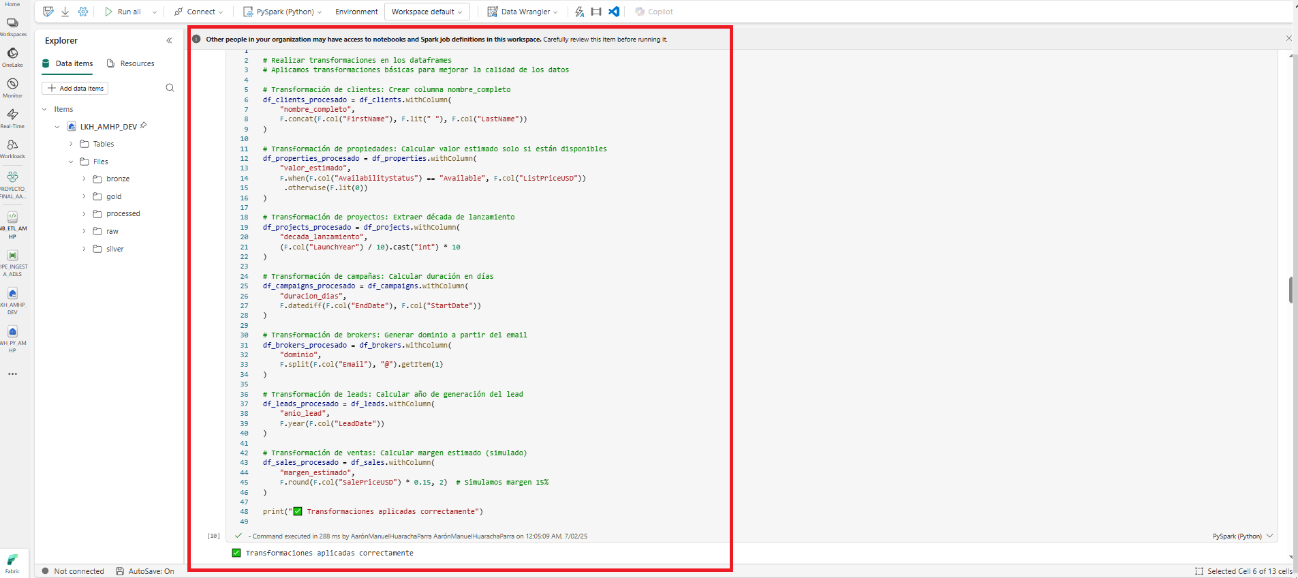
Este bloque crea la estructura de carpetas dentro del Lakehouse utilizando mssparkutils.fs.mkdirs, siguiendo la arquitectura medallion (bronze, silver, gold). Esta organización permite controlar y rastrear el ciclo de vida de los datos desde su ingesta hasta su análisis final, dentro de un entorno controlado y bien estructurado.

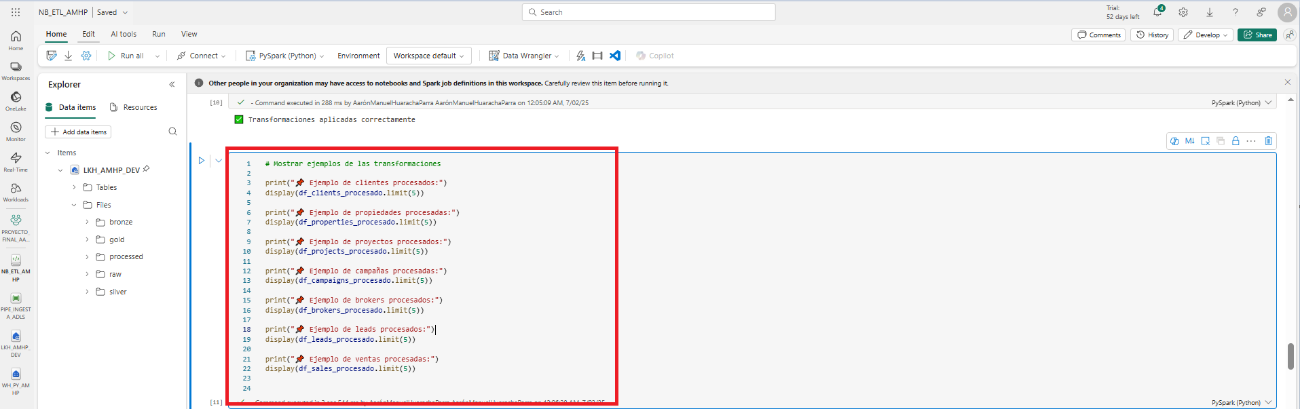
Este bloque define esquemas explícitos (StructType) para cada conjunto de datos que será cargado con PySpark, lo que permite controlar con precisión los tipos de datos de cada columna (como IntegerType, StringType y DateType). Al aplicar estos esquemas al momento de leer los archivos CSV desde el Lakehouse, se garantiza la calidad e integridad de los datos, evitando errores por inferencias automáticas incorrectas. Además, estos esquemas facilitan la validación, transformación y posterior modelado de los datos (como en un modelo estrella), asegurando que cada entidad del dominio (brokers, campaigns, clients, leads, projects, properties y sales) esté correctamente estructurada desde la fase de ingesta.

Este bloque lee archivos CSV desde la carpeta Raw del Lakehouse utilizando PySpark, aplicando los esquemas definidos previamente para garantizar la correcta estructura de los datos. Se usa el formato "csv" con la opción "header" activada y se especifica la ruta de cada archivo por entidad (brokers, campaigns, clients, leads, projects, properties y sales). El bloque try-except permite manejar errores durante la lectura y asegurar que todos los archivos se carguen correctamente en sus respectivos DataFrames.

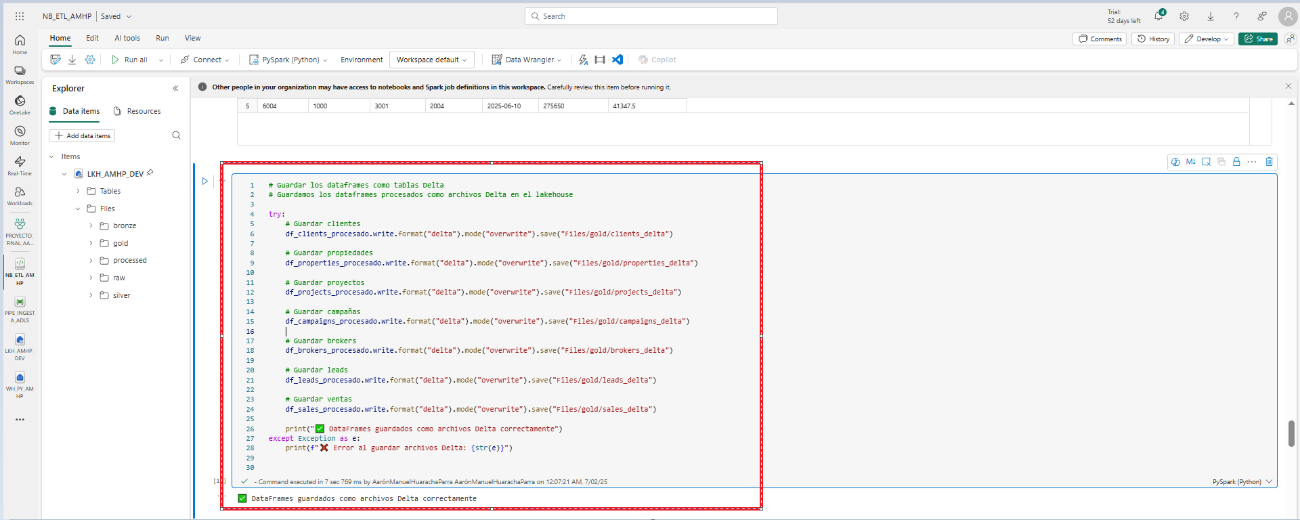
Este bloque realiza una exploración inicial de los datos cargados mostrando una vista previa (primeras 5 filas) de cada DataFrame correspondiente a las distintas entidades (brokers, campaigns, clients, leads, projects, properties y sales). Se utiliza la función display() para visualizar los datos de forma tabular en el entorno del notebook, lo cual permite verificar rápidamente que la carga fue exitosa y que los datos tienen la estructura esperada.

Este bloque muestra información básica de cada DataFrame: la cantidad total de registros (count()) y la estructura de columnas con sus tipos de datos (printSchema()). Esto permite validar que los datos fueron correctamente leídos, que no hay columnas faltantes y que los tipos coinciden con los definidos en los esquemas. Es una práctica clave antes de realizar transformaciones o análisis.

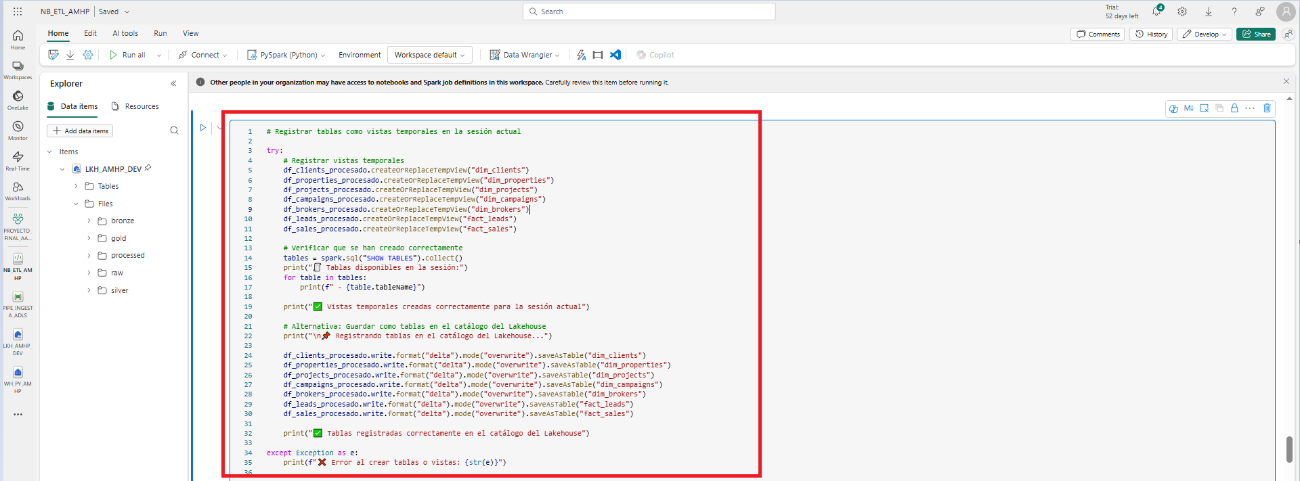
Este bloque aplica transformaciones básicas a los DataFrames para mejorar la calidad y enriquecer los datos. Se crean nuevas columnas derivadas: nombre\_completo en clientes (concatenando nombre y apellido), valor\_estimado en propiedades (basado en disponibilidad), decada\_lanzamiento en proyectos (calculando la década), duracion\_dias en campañas (diferencia entre fechas), dominio en brokers (extraído del correo), año\_lead en leads (a partir de la fecha del lead), y margen\_estimado en ventas (calculado como el 15% del precio). Estas transformaciones preparan los datos para análisis más profundos y su posterior modelado.

Este bloque muestra una vista previa de las transformaciones aplicadas a cada DataFrame procesado, utilizando display() para visualizar las primeras 5 filas. Esto permite verificar que las nuevas columnas calculadas (como nombre\_completo, valor\_estimado, decada\_lanzamiento, etc.) se han generado correctamente y que los datos procesados están listos para ser almacenados o utilizados en el análisis.

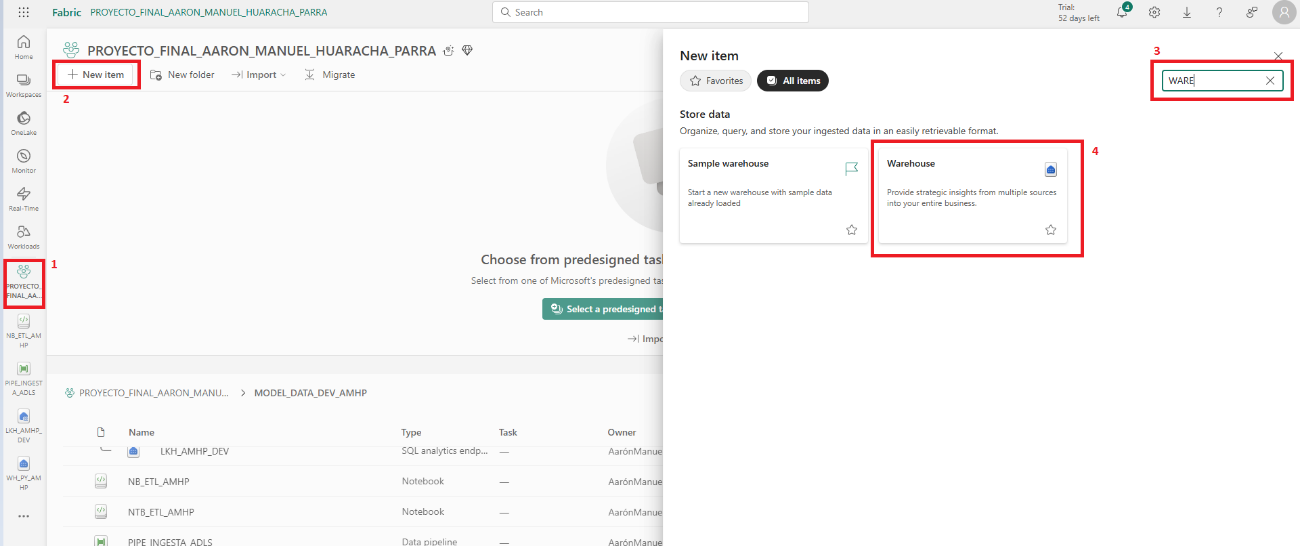
Este bloque guarda los DataFrames transformados como archivos Delta Lake en la carpeta gold del Lakehouse, utilizando el formato "delta" y el modo "overwrite" para sobrescribir versiones anteriores. Cada dataset procesado (clientes, propiedades, proyectos, campañas, brokers, leads y ventas) se almacena en su ruta correspondiente (Files/gold/...). Esto permite conservar los datos en un formato optimizado para análisis, con soporte para transacciones, versiones y consultas eficientes. El bloque try-except controla errores durante el guardado, asegurando una carga limpia en caso de éxito.

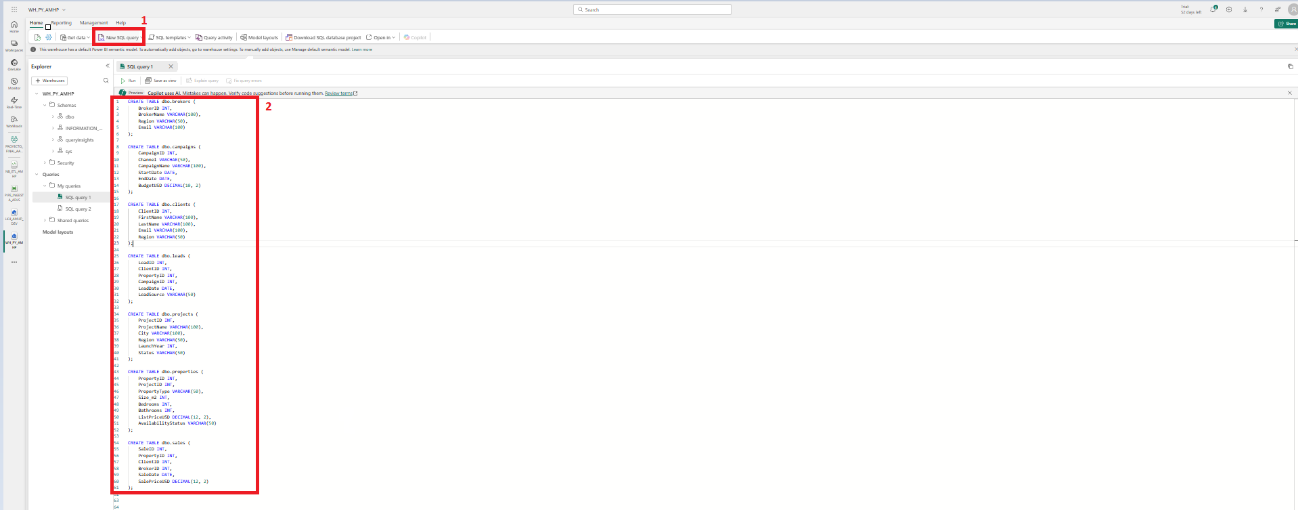


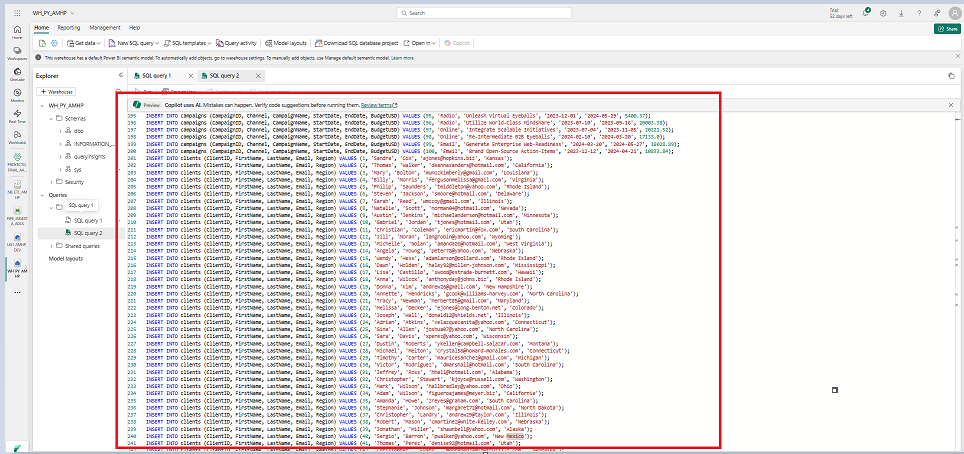
Este bloque registra los DataFrames transformados como vistas temporales dentro de la sesión activa de Spark, usando createOrReplaceTempView(), lo que permite consultarlos directamente con SQL sin necesidad de almacenarlos físicamente. Luego, como paso adicional, también se registran como tablas permanentes en el catálogo del Lakehouse mediante saveAsTable(), lo que garantiza su persistencia y disponibilidad para otros usuarios o notebooks. Se utiliza un bloque try-except para manejar errores y se valida que las vistas temporales hayan sido creadas correctamente usando SHOW TABLES. Esto es esencial para habilitar consultas SQL sobre las dimensiones y hechos, tanto en tiempo real como desde el catálogo.



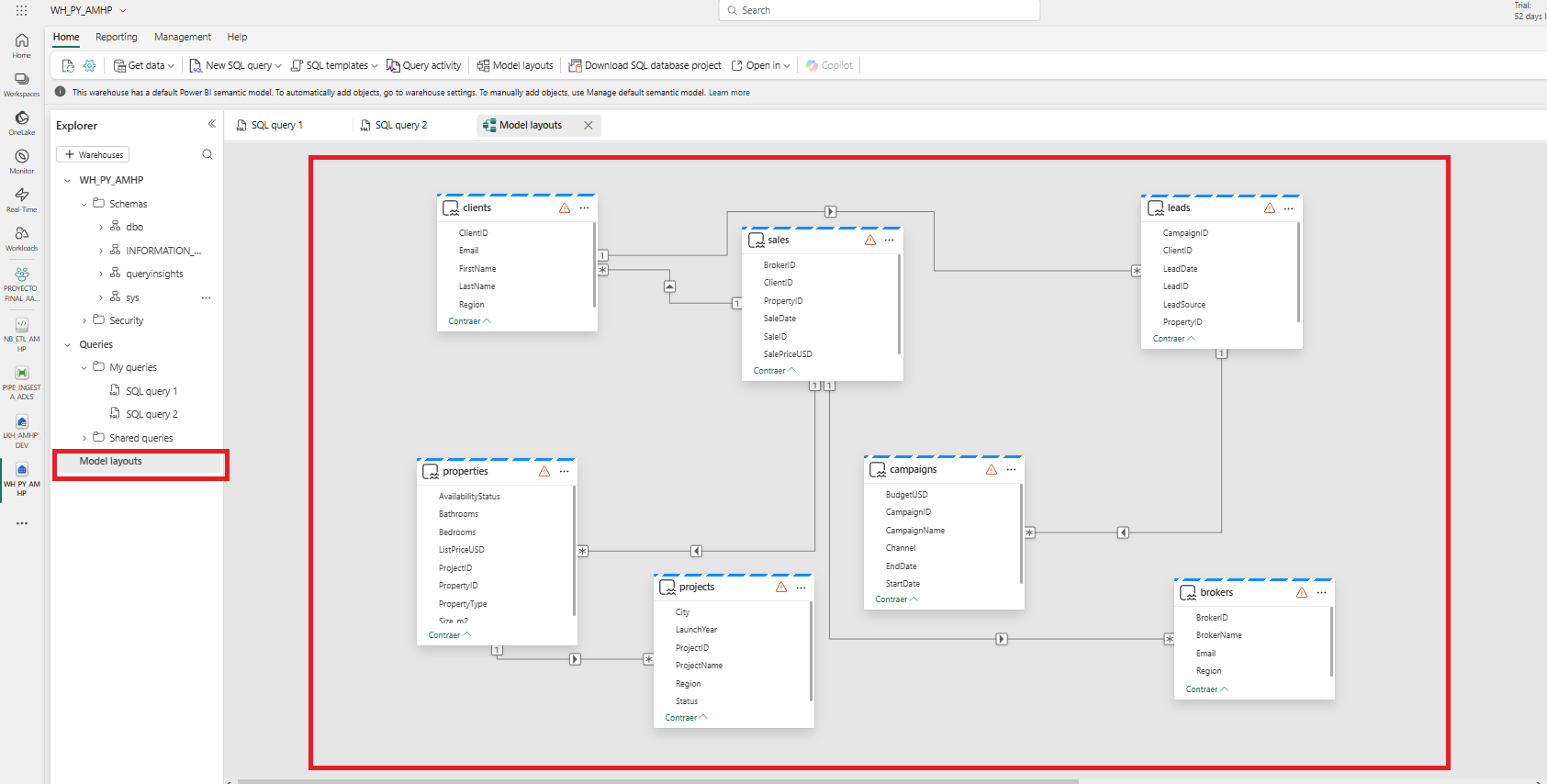
A continuación, crearemos el modelo semántico ('Semantic Model'), iniciando con la creación del Warehouse que servirá como base para su diseño y consulta.



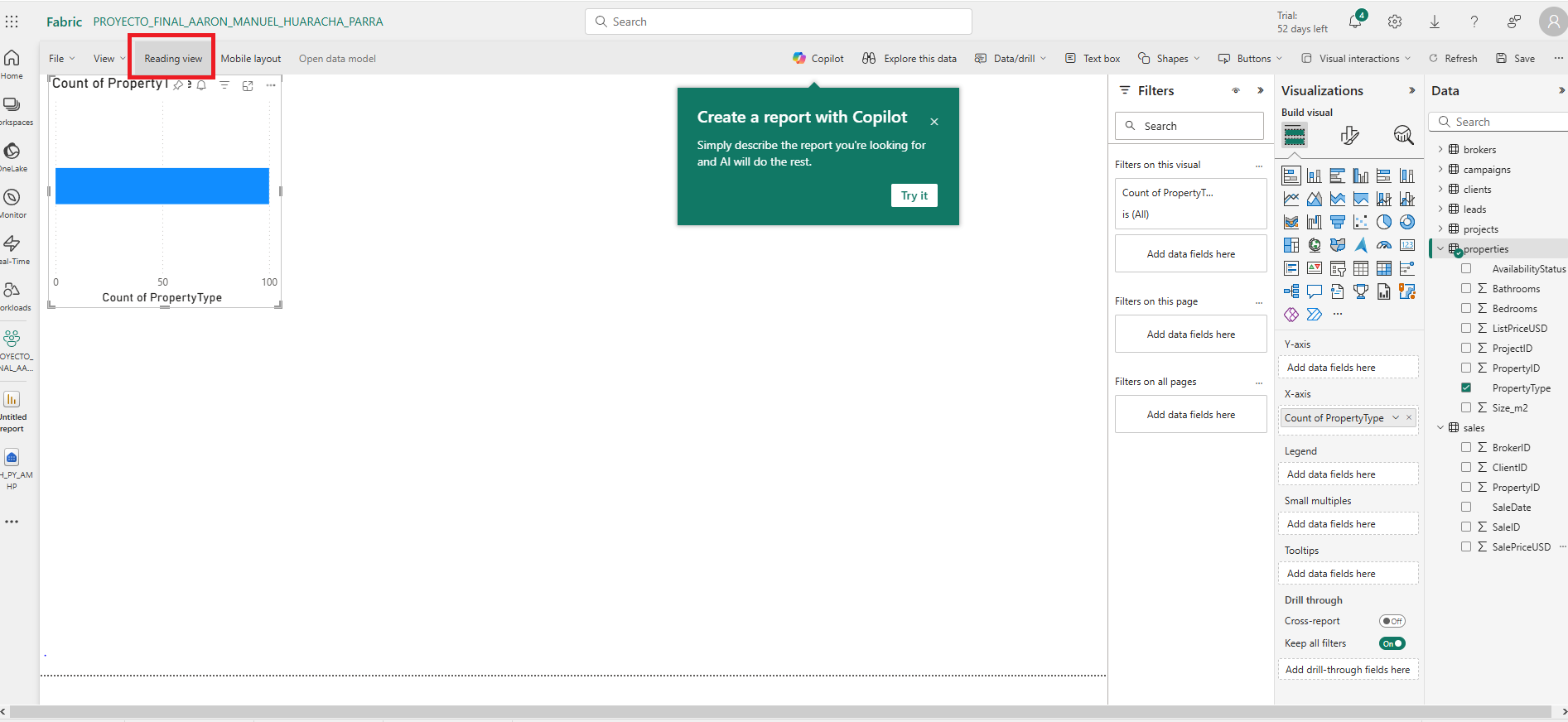
Procedemos a generar una nueva consulta dentro del Warehouse, en la cual definiremos las tablas mediante instrucciones SQL, estableciendo así la base estructural de nuestro modelo semántico.

Una vez creada las tablas, le vamos a insertar registros a todas las tablas que hemos creado, por ejemplo.

Este modelo semántico representa un modelo estrella, donde las tablas sales y leads actúan como tablas de hechos, conectadas a dimensiones como clients, brokers, campaigns, properties y projects mediante claves foráneas. Esta estructura optimiza el análisis en Power BI, facilitando consultas rápidas y relaciones claras entre entidades del negocio inmobiliario.

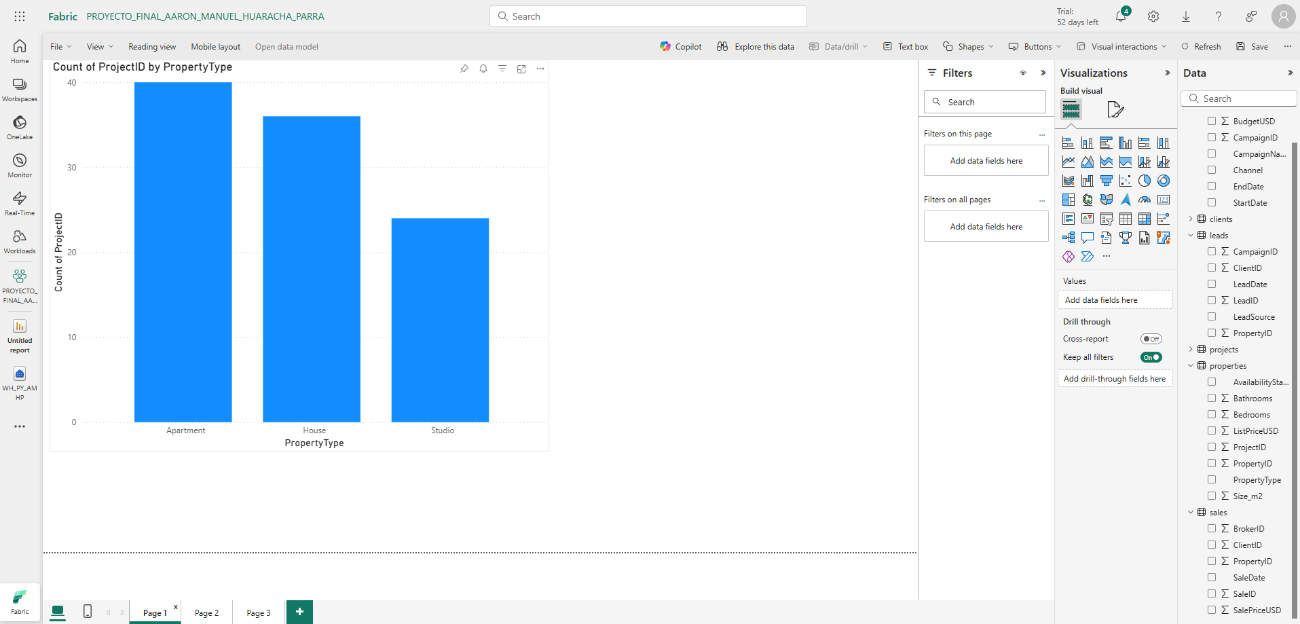


Generamos un reporte visual para identificar patrones y detectar posibles incidencias en los datos.

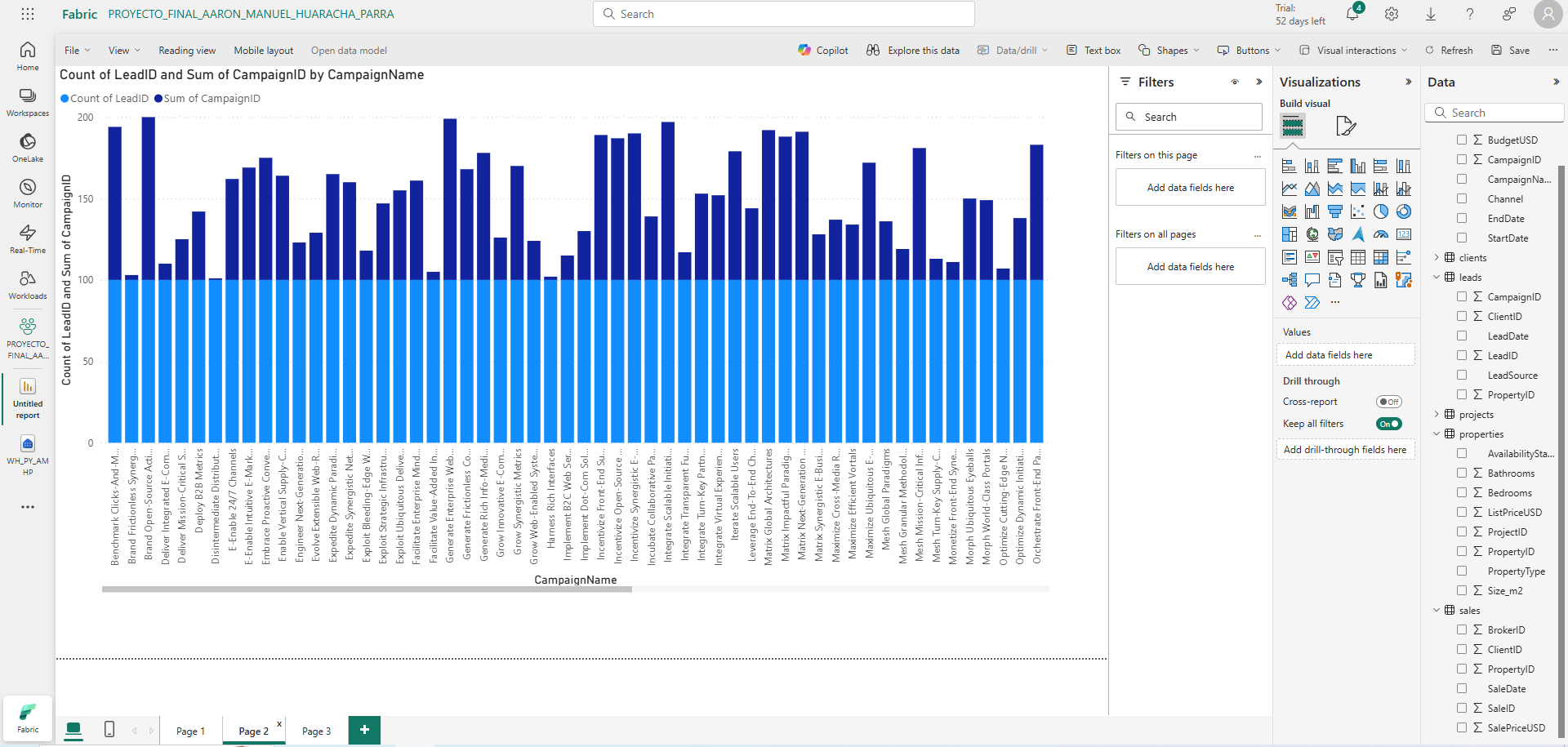


Este modelo semántico representa una estructura de tipo estrella, diseñada para análisis en el sector inmobiliario. Las tablas sales y leads funcionan como tablas de hechos, donde se almacenan eventos clave como ventas y registros de prospectos. Estas se conectan mediante claves foráneas a diversas tablas de dimensión, como:

* clients (información del cliente),
* brokers (agentes o corredores),
* campaigns (campañas de marketing),
* properties (detalle de las propiedades ofertadas),
* projects (proyectos o desarrollos inmobiliarios).

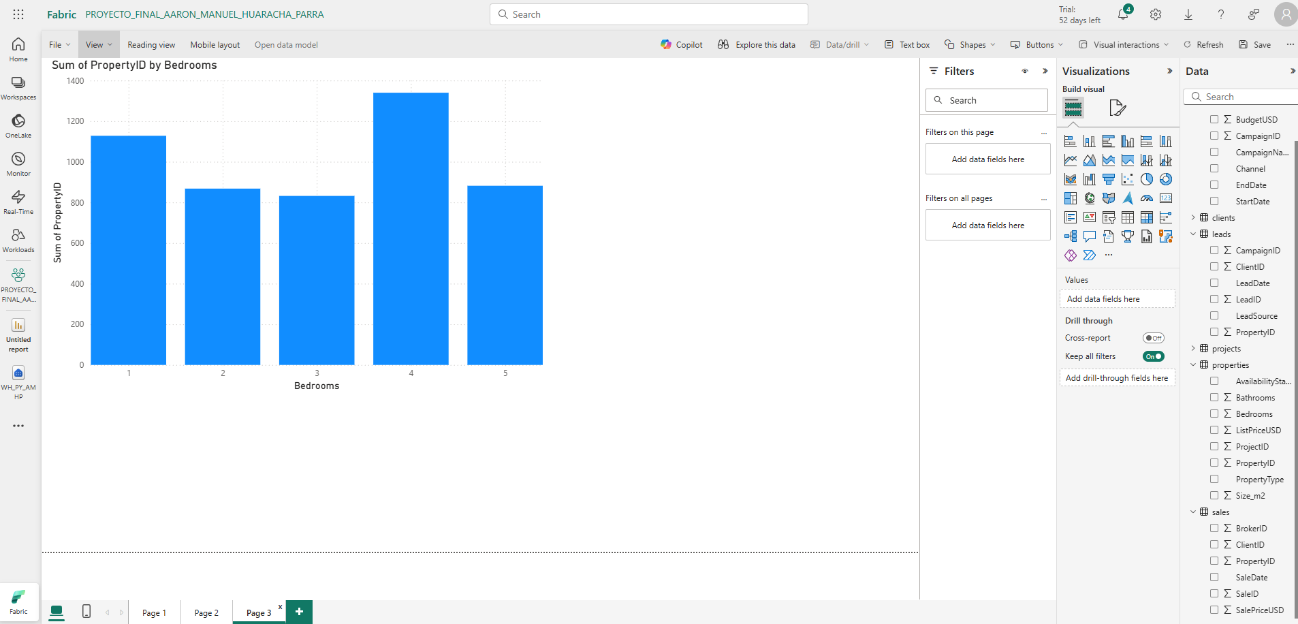
Esta estructura facilita relaciones claras entre entidades clave del negocio y permite un análisis eficiente en Power BI, mediante visualizaciones como conteos por tipo de propiedad, análisis de ventas, segmentaciones de clientes y rendimiento de campañas.

Esta visualización representa un análisis comparativo del rendimiento de campañas de marketing dentro del modelo semántico. Se grafican dos métricas clave agrupadas por el campo CampaignName: el conteo de leads (LeadID) y la suma de identificadores de campaña (CampaignID).

Aunque el conteo de leads permite identificar qué campañas generaron mayor atracción de prospectos, la suma de CampaignID carece de valor analítico directo, ya que representa un identificador repetido. La estructura refleja el uso de la tabla de hechos leads conectada a la dimensión campaigns, lo que permite un análisis detallado del impacto de cada campaña. Este enfoque apoya la toma de decisiones estratégicas en marketing y optimiza el seguimiento del desempeño comercial en el sector inmobiliario.

Esta visualización muestra la suma de PropertyID agrupada por el número de dormitorios (Bedrooms) de las propiedades. Aunque PropertyID es un identificador único, su suma no representa un valor analítico real; sin embargo, esta gráfica permite observar la distribución relativa de propiedades según la cantidad de dormitorios.

El análisis se realiza sobre la tabla de dimensión properties, conectada al modelo estrella. Se observa que las propiedades con 4 dormitorios son las más frecuentes, seguidas por aquellas con 1 dormitorio. Este tipo de visualización apoya la comprensión de la oferta inmobiliaria disponible, permitiendo tomar decisiones orientadas a la demanda del mercado.



**4. REFLEXIONES: DIFICULTADES, BUENAS PRÁCTICAS, “PRÓXIMOS PASOS”.**

**Dificultades**

* Integración inicial con ADLS requería configuración precisa de permisos.
* Lectura de archivos CSV con estructuras inconsistentes forzó la definición manual de esquemas.
* Al transformar múltiples datasets en paralelo, fue clave validar integridad referencial entre hechos y dimensiones.

**Buenas prácticas**

* Uso de esquemas explícitos en PySpark para control de calidad de datos.
* Aplicación del modelo Bronze/Silver/Gold para trazabilidad.
* Registro de tablas como vistas temporales y luego permanentes desde notebooks, lo cual facilitó SQL y modelado.
* Separación clara entre transformaciones técnicas y visualizaciones de negocio.

**Próximos pasos**

* Incorporar streaming de eventos en tiempo real (por ejemplo, nuevos leads desde portales web).
* Automatización de alertas ante fallos de pipeline mediante Fabric Capacity Metrics App.
* Ampliación del modelo con datos externos: indicadores macroeconómicos, tasas hipotecarias, tipo de cambio.
* Optimización de Power BI con KPIs personalizados y segmentaciones avanzadas para el equipo comercial.