PROYECTO ETL USANDO AZURE DATA FACTORY

Presentado por: Rafael Chamorro, Gustavo Adolfo Escobar, Elizabeth Obando Galíndez y Sebastián Hidalgo Marín

INTRODUCCIÓN

Este documento describe el desarrollo del proyecto de integración y transformación de datos (ETL) utilizando Azure Data Factory (ADF) como herramienta para coordinar los procesos ETL. El objetivo es consolidar la información proveniente de tres fuentes heterogéneas:

- Base de datos PostgreSQL (Neon).
- Colecciones en MongoDB Atlas.
- Archivos CSV almacenados en un Data Lake de Azure.

A partir de estas fuentes, se construye un dataset unificado y consistente, que resulta de la realización de actividades como control de calidad, manejo de valores faltantes, estandarización de variables y diseño de pipelines reutilizables y escalables.

ACTIVIDADES REALIZADAS.

1. Se hizo el despliegue de los siguientes recursos en Azure:
   a. Grupo de recursos: contenedor lógico para agrupar todos los componentes del proyecto.
   b. Azure Data Lake Storage: repositorio centralizado de archivos.
   c. Azure Data Factory (ADF): servicio de integración de datos para gestionar los flujos de datos o administrar los pipelines ETL. Dentro del ADF se configuraron:
      - Linked Services hacia MongoDB Atlas, Neon y Data Lake de Azure.
      - Datasets que representan tablas, colecciones y archivos CSV.
      - Pipelines y Data Flows para extraer, transformar y cargar la información en el Data Lake y en las salidas finales del proyecto.

2. Se diseñó un pipeline central (RecursoFinal) encargado de realizar el *merge* de todas las fuentes de datos a través de tres grandes bloques:

   a. Extracción y transformación desde PostgreSQL (Neon).

   Se creó un dataset asociado a la conexión Neon Conn y se definió una consulta SQL cuyo propósito fue consolidar, en una sola estructura usando PID como llave. La información procedente de varias tablas:

   - amesdbtemp (tabla principal de propiedades).
   - floordetail (detalle de pisos, baños y habitaciones).
   - saleproperty (datos de venta).
   - mszoning (zonificación).
   - typequality (tabla de códigos de calidad usada con distintos alias).
   - El vínculo clave para integrar las tablas es el campo PID, que identifica cada propiedad.

   En la consulta se realizaron acciones como:

   - Selección de las variables principales de cada tabla y reemplazo de identificadores numéricos por códigos o descripciones legibles a partir de typequality.
   - Cálculo de métricas agregadas como la construcción de GrLivArea como suma de áreas de piso (1st Flr SF, 2nd Flr SF, Low Qual Fin SF), usando COALESCE para tratar valores nulos como 0.

- Enriquecimiento con datos de venta. De saleproperty se incorporan, por propiedad: Fecha máxima de venta y su descomposición en mes y año (MoSold, YrSold), tipo y condición de venta y SalePrice.
- Asegurarse que los valores agregados sean enteros. Se presta especial atención a la conversión de tipos de datos.

b. Procesamiento del archivo AMES Properties almacenado en el Data Lake.

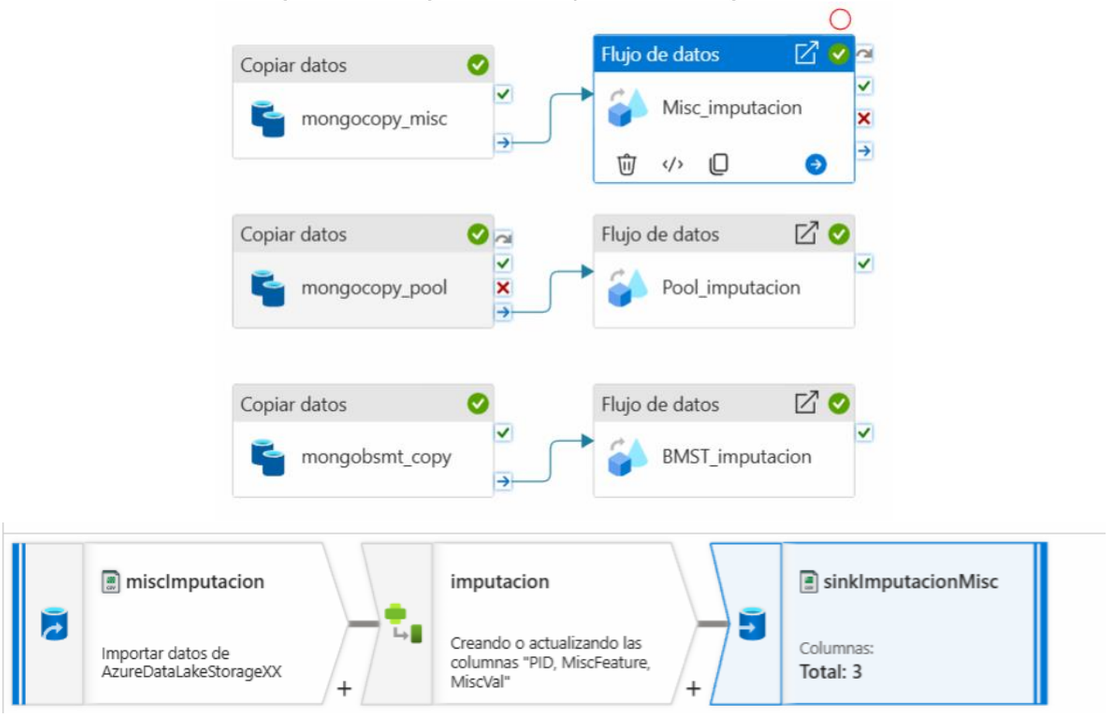
Este dataset se leyó desde un archivo CSV, ajustando el delimitador para que coincidiera con el formato real del archivo. Durante la carga se detectó un problema en la última columna, cuyos valores venían con ",,". Para corregirlo eficientemente, se creó una columna derivada donde mediante una expresión se eliminó las comas sobrantes, imputo valores vacíos de la columna “Year Remod/add” y convirtió en entero.

c. Extracción, normalización e imputación de datos desde MongoDB Atlas.

Los datos de MongoDB Atlas se exportaron a cuatro archivos CSV (uno por colección) mediante un pipeline en ADF (Mongo Copy to CSV) que copió cada colección a una carpeta específica en el Data Lake. Sobre esos archivos se construyó un Data Flow (ver figura 1) que:

- En la “columna derivada”, se utilizó la opción de “patrón de columna” para que el flujo tomase automáticamente todas las columnas disponibles, sin tener que seleccionarlas una por una.
- Asignó tipos de datos y aplicó reglas de imputación diferenciadas para variables numéricas y categóricas.
- Reemplazó nulos, NaN y vacíos por 0 en variables numéricas y por "NA" en variables de texto.

Figura 1. Vista general del Pipeline de Mongo en ADF.



3. Procesamiento para consolidación final del dataset: resultados del proceso ETL.

a. Extracción de datasets procesados y estandarización.

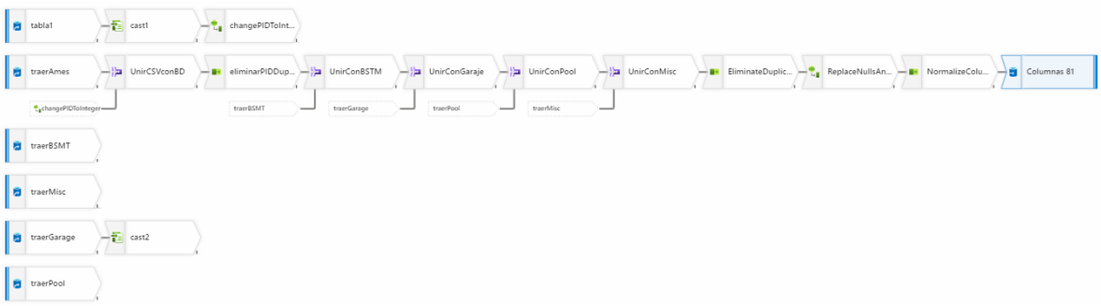
Tras el preprocesamiento de las diferentes fuentes, los datos fueron extraídos e incorporados al flujo, corrigiendo en el mismo proceso las discrepancias de tipos de datos para asegurar una integración limpia y consistente.

b. Uniones (JOIN) entre los datos.

Cuando los datos estuvieron listos, se inició la unión por extremo izquierdo de los mismos usando el PID como identificador principal de las columnas. La unión sigue el orden a continuación:

- Se unió la base de datos “amesCSV” con la BD Relacional Postgres alojada en NeonConn, dado que estas eran las que tenían mayor volumen de datos en relación con las demás.
- Con la base de datos de propiedades ya única, se prosigue con la unión de las bases de datos de MongoDB, empezando con BSTM y siguiendo con Garage, Pool y Misc.
- Una vez unidas las bases de datos, se eliminan las columnas duplicadas por PID que viene de las distintas tablas, puesto que ADF no elimina columnas al hacer los JOIN para asegurar la integridad de los datos
- Al unir las tablas, también se generan valores nulos, pues hay columnas que no son comunes entre los datos. Ante esto, se procede con el reemplazo de los valores nulos usando “NA” en las columnas categóricas y 0 en las numéricas.
- Después se realiza la normalización de las columnas eliminando espacios y caracteres especiales usando *regex.*
- Para finalizar, se exporta la base de datos final usando un CSV para su evaluación.

Figura 2. Integración final de las fuentes de datos.



Como resultado de la orquestación de estos componentes se obtuvo la consolidación de un dataset enriquecido que integró:

- Información estructural y de calidad de las propiedades (Neon/PostgreSQL).
- Atributos adicionales provenientes del archivo AMES Properties (CSV en Data Lake).
- Variables complementarias estandarizadas desde MongoDB Atlas.

Se resolvieron problemas de:

- Delimitadores inconsistentes en CSV.
- Columnas con caracteres residuales (',,').
- Valores faltantes, NaN y campos vacíos en distintas fuentes.

Finalmente, se logró un esquema consistente y documentado, apto para análisis exploratorio, modelado estadístico y aplicación de algoritmos de aprendizaje automático.

4. Integración final con GitHub para revisión.

Como último paso, se configuró la integración de Azure Data Factory con un repositorio en GitHub, donde se almacenaron el código de los pipelines, los datasets y la configuración asociada al proyecto. Este repositorio fue compartido con un experto evaluador, de modo que pueda inspeccionar directamente la implementación, verificar la trazabilidad de las transformaciones y comprobar la coherencia y consistencia del flujo de trabajo con los resultados esperados para la calificación.

LECCIONES APRENDIDAS.

Durante el desarrollo del proyecto se identificaron varias lecciones clave desde la perspectiva de la ingeniería y la ciencia de datos:

1. Importancia de diseñar procesos recursivos y reutilizables en ADF. La construcción de pipelines parametrizables y Data Flow genéricos (por ejemplo, uso de patrones de columnas) permite:

    - Reducir el número de actividades específicas.
    - Aumentar la eficiencia y mantenibilidad.
    - Reutilizar la misma lógica para nuevas fuentes o esquemas similares.
    - Escalar el proceso a futuros proyectos sin rediseñar todo desde cero.

2. Uso de expresiones y funciones avanzadas**.** El aprovechamiento de funciones como COALESCE, expresiones condicionales (iif), manejo de NaN, y el empleo de expresiones anidadas en columnas derivadas, permite resolver múltiples problemas de calidad de datos en un solo paso, reduciendo la complejidad del flujo.

3. Búsqueda de soluciones específicas por tipo de fuente. Cada origen (PostgreSQL, CSV, Mongo) presenta desafíos propios:

    - En bases relacionales, el énfasis está en consultas SQL eficientes, joins y agregaciones consistentes.
    - En archivos CSV, el foco está en el parseo correcto, manejo de delimitadores y limpieza de cadenas.
    - En Mongo, el reto principal es normalizar esquemas flexibles y asegurar una imputación robusta. En este contexto fue útil recurrir a funciones como COALESCE.

4. Necesidad de pruebas iterativas. Antes de consolidar un pipeline, es fundamental hacer pruebas de funcionamiento parciales:

    - Validar subconjuntos de datos.
    - Revisar tipos de datos y resultados de transformaciones.
    - Confirmar que las reglas de imputación no distorsionan los datasets.

5. Trabajo colaborativo en Azure Data Factory. Todos los integrantes del equipo participaron activamente en la construcción y prueba de los pipelines, hecho que:

    - Evitó la concentración del conocimiento en pocas personas.
    - Facilitó la revisión cruzada de expresiones, consultas y configuraciones.
    - Mostró el potencial colaborativo de ADF, aprovechando la edición simultánea y la validación en tiempo real.

6. Gestión de la calidad de datos como prioridad**.** Más allá de extraer y cargar información, el proyecto evidenció que:

    - La identificación temprana de problemas (nulos, formatos inconsistentes, códigos sin descripción) evita errores en etapas posteriores de análisis.