Informe Final del Proceso ETL – Proyecto BD Ames

Equipo: E3 CTRL Z
Miguel Jaramillo
Luisa Castaño
Sebastían Correa
Valentina Arana

1. Introducción

El proyecto tiene como objetivo construir un proceso ETL completo en Azure Data Factory (ADF) que integre datos de distintas fuentes de la base de datos Ames Housing (relacional y no relacional) para generar un archivo de salida consolidado llamado salida.csv.

El proceso abarca desde la ingesta de datos crudos hasta la generación de un dataset consolidado llamado salida.csv, implementando transformaciones de datos, cálculo de métricas agregadas, tratamiento de valores faltantes, y consolidación por propiedad (PID).

2. Objetivo

- Construir un pipeline maestro en ADF que orqueste dependencias secuenciales entre los data flow.
- Obtener un archivo salida.csv con 81 columnas estandarizadas, preparado para análisis.
- Aplicar reglas de negocio: cálculo de GrLivArea, totales de baños/habitaciones, normalización de fechas (MoSold/YrSold) y manejo de nulos.

3. Arquitectura y Estrategia de Datos

La arquitectura se diseñó para minimizar la carga en la nube y manejar las dependencias entre fuentes no estructuradas y relacionales:

3.1. Fuente Relacional (PostgreSQL/Neon): Para evitar la sobrecarga de transferir tablas crudas masivas a la nube, se optó por realizar la extracción y transformación inicial de manera externa utilizando DBeaver.

Se ejecutó una consulta SQL compleja directamente en DBeaver conectada a Neon, así ésta consulta calculó métricas clave (como GrLivArea), normalizó fechas (MoSold, YrSold) y realizó la agregación compleja de habitaciones y baños (GROUP BY PID) desde la tabla FloorDetail. Finalmente se exportó un único archivo CSV optimizado que fue cargado manualmente al Data Lake.

3.2. Fuente NoSQL (MongoDB):

La integración de las colecciones provenientes de MongoDB se llevó a cabo mediante un pipeline de extracción dedicado dentro de Azure Data Factory. Este proceso se estructuró en dos etapas principales:

- Fase de ingesta: Se implementaron actividades de Copy Data destinadas a replicar las colecciones crudas desde MongoDB Atlas hacia el Data Lake. Este enfoque aseguró que las estructuras originales de los documentos JSON quedaran almacenadas sin alteración, preservando su integridad para posteriores transformaciones.
- Preprocesamiento y normalización: Posteriormente, un Data Flow especializado se encargó de tomar estos archivos y unificar los documentos en una estructura coherente. El resultado de este Data Flow es un dataset intermedio depurado y estandarizado, adecuado para participar en la fase de integración final con las otras fuentes del proyecto.

4. Orquestación: Pipeline Master en ADF

El pipeline maestro (PL_Master) coordina la ejecución secuencial de los diferentes componentes del flujo completo:

- Procesamiento de datos: Se ejecuta primero el Data Flow encargado de la integración de las colecciones de MongoDB, generando un dataset intermedio unificado.

- Integración final: Una vez disponible la información de Mongo, se activa el Data Flow que consolida los datos provenientes de las tres fuentes: el CSV resultante del procesamiento SQL, el archivo AmesProperty.csv y el dataset intermedio de MongoDB.

Este Data Flow final emplea uniones Join basadas en el campo PID y aplica reglas adicionales de limpieza mediante transformaciones de tipo Select y Derived Column.

5. Aseguramiento de calidad y validación en Python

Durante el proceso se identificaron inconsistencias entre las fuentes, especialmente diferencias en tipos de datos y representaciones numéricas en códigos categóricos. Para resolverlas, se implementó un Notebook en Python como etapa final de validación y ajuste.

En esta fase se corrigieron discrepancias en variables categóricas como MS Zoning y calidades, se verificó la integridad y correspondencia de datos procedentes de MongoDB y se aplicaron reglas de negocio finales, entre ellas:

- Cálculo definitivo de GrLivArea como suma de 1st Flr SF, 2nd Flr SF y Low Qual Fin SF.
- Manejo estandarizado de valores nulos, asignando 0 a variables numéricas y “NA” a variables categóricas.
- Ajuste de YearRemodAdd, imputando YearBuilt cuando fuese necesario.

6. Resultados

El proceso completo produjo el archivo salida.csv, ubicado en el contenedor output del Data Lake.

- Dimensiones: 2930 filas y 81 columnas.
- Validación: El archivo cumplió satisfactoriamente con los criterios establecidos:
    - Ausencia total de valores nulos.
    - Estructura de columnas exacta.
    - Tipos de datos coherentes con las especificaciones del proyecto.

Logrando un porcentaje de éxito del 90%:

```
#Final Report
print("\n==== VALIDATION COMPLETE ====")
print("Total tests:", total_tests)
print("Tests passed:", passed_tests)
print("Percentage passed:", round(passed_tests/total_tests * 100, 2), "%")
```

```
==== VALIDATION COMPLETE ====
Total tests: 20
Tests passed: 18
Percentage passed: 90.0 %
```

7. Lecciones aprendidas

El desarrollo del proyecto permitió identificar aspectos clave:

- La importancia de estandarizar el tipo de dato de las llaves primarias antes de realizar uniones, ya que discrepancias entre enteros y cadenas pueden ocasionar pérdida de registros en Data Flows.
- La conveniencia de emplear estrategias ELT/ETL híbridas, delegando transformaciones pesadas al motor SQL para mejorar el rendimiento global.
- La utilidad de complementar ADF con herramientas externas como Python para tareas avanzadas de control de calidad o validaciones lógicas complejas.

8. Conclusiones:

El proyecto logró integrar de manera efectiva datos provenientes de fuentes relacionales, documentales y archivos planos. La combinación de SQL, Azure Data Factory y Python permitió construir una arquitectura robusta, optimizada y capaz de cumplir con los requisitos estrictos del formato de salida.

La solución consolidada demuestra la importancia de seleccionar adecuadamente la estrategia de transformación en función de la naturaleza de las fuentes y de los objetivos analíticos del sistema.

9. Anexos:

Link del repositorio: https://github.com/MCD-Infrati/etlproject2025-2-e3-ctrlz