PIPELINE ETL PARA ANÁLISIS DE PROPIEDADES INMOBILIARIAS CON AZURE DATA FACTORY

Equipo E3 - Ctrlz
Miguel Jaramillo, Luisa Castaño
Sebastián Correa y Valentina Arana

CONTENIDO

OBJETIVO

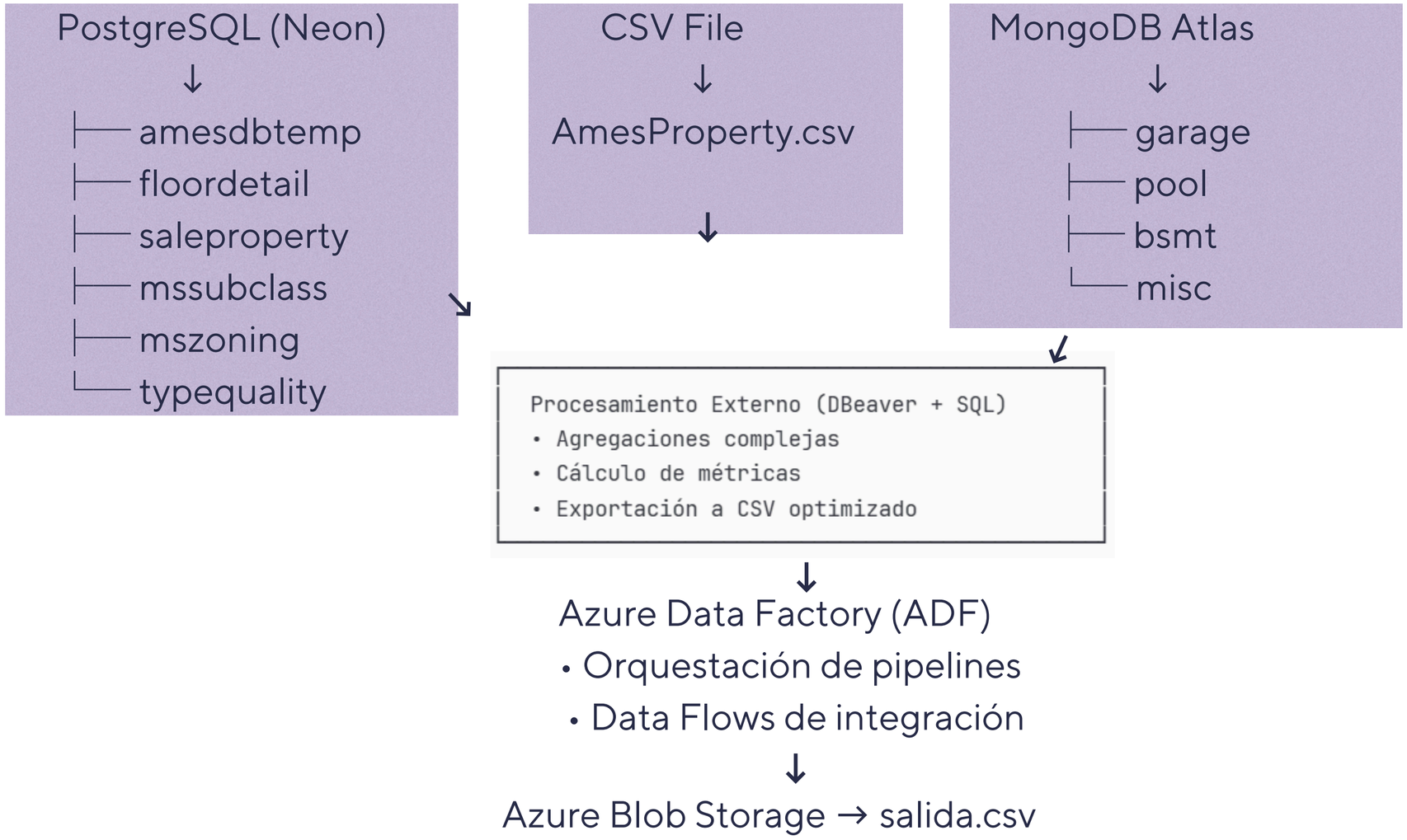
Diseñar e implementar un proceso ETL completo que integre datos de múltiples fuentes para generar un dataset unificado del Ames Housing Dataset.

Reto Principal: Consolidar información de 2,930 propiedades desde:

- 6 tablas relacionales (Azure SQL)
- 1 archivo CSV
- 4 colecciones NoSQL (MongoDB Atlas)

Entregable Final: Archivo salida.csv con 81 características por propiedad

Arquitectura de Fuentes de Datos



Estrategia Híbrida: ETL + ELT

¿Por qué procesamiento externo con DBeaver?
Desafío: Tablas masivas en PostgreSQL/Neon
Solución: Realizar transformaciones pesadas en la fuente

Proceso:

1 Extracción + Transformación en DBeaver
- Query SQL compleja ejecutada en Neon
- Cálculo de GrLivArea, agregaciones de baños/habitaciones
- GROUP BY PID para consolidar FloorDetail
- Exportación a CSV optimizado

2 Carga manual al Data Lake de Azure
- CSV pre-procesado listo para integración

3 Orquestación en ADF
- Pipeline maestro (PL_Master)
- Integración con MongoDB y AmesProperty.csv

Beneficio: Reducción de carga en la nube + Mejor rendimiento

Proceso de Extracción

PostgreSQL/Neon (Procesamiento Externo)

- 6 tablas relacionales
- Conexión: DBeaver a Neon
- Método: Query SQL compleja + Exportación CSV
- Transformaciones aplicadas en origen
- Datos: Características agregadas por PID

Archivo CSV (AmesProperty.csv)

- Información básica de propiedades
- Ubicación: Azure Blob Storage
- Método: DelimitedText Dataset
- Datos: ID, ubicación, año construcción

MongoDB Atlas (4 colecciones)

- garage: Información de garajes
- pool: Características de piscinas
- bsmt: Detalles de sótanos
- misc: Características misceláneas
- Método: MongoDB Connector + Lookup Activities

Aseguramiento de Calidad con Python

Desafío Identificado:

Inconsistencias entre fuentes (tipos de datos, códigos categóricos)

Solución: Notebook Python como etapa final

Correcciones Aplicadas:
✓ Variables categóricas (MS Zoning, calidades)
✓ Integridad de datos de MongoDB
✓ Reglas de negocio finales:
• GrLivArea = 1stFlrSF + 2ndFlrSF + LowQualFinSF
• Manejo de nulos: 0 (numéricos) / "NA" (categóricos)
• YearRemodAdd: imputación con YearBuilt

Resultado:
✅ 18 de 20 tests pasados (90% de éxito)
✅ 0 valores nulos en output final

Decisiones Técnicas

¿Por qué Azure Data Factory?

✅ Integración nativa con servicios Azure
✅ Soporte multi-fuente (SQL, NoSQL, archivos)
✅ Orquestación visual de pipelines
✅ Escalabilidad y monitoreo integrado

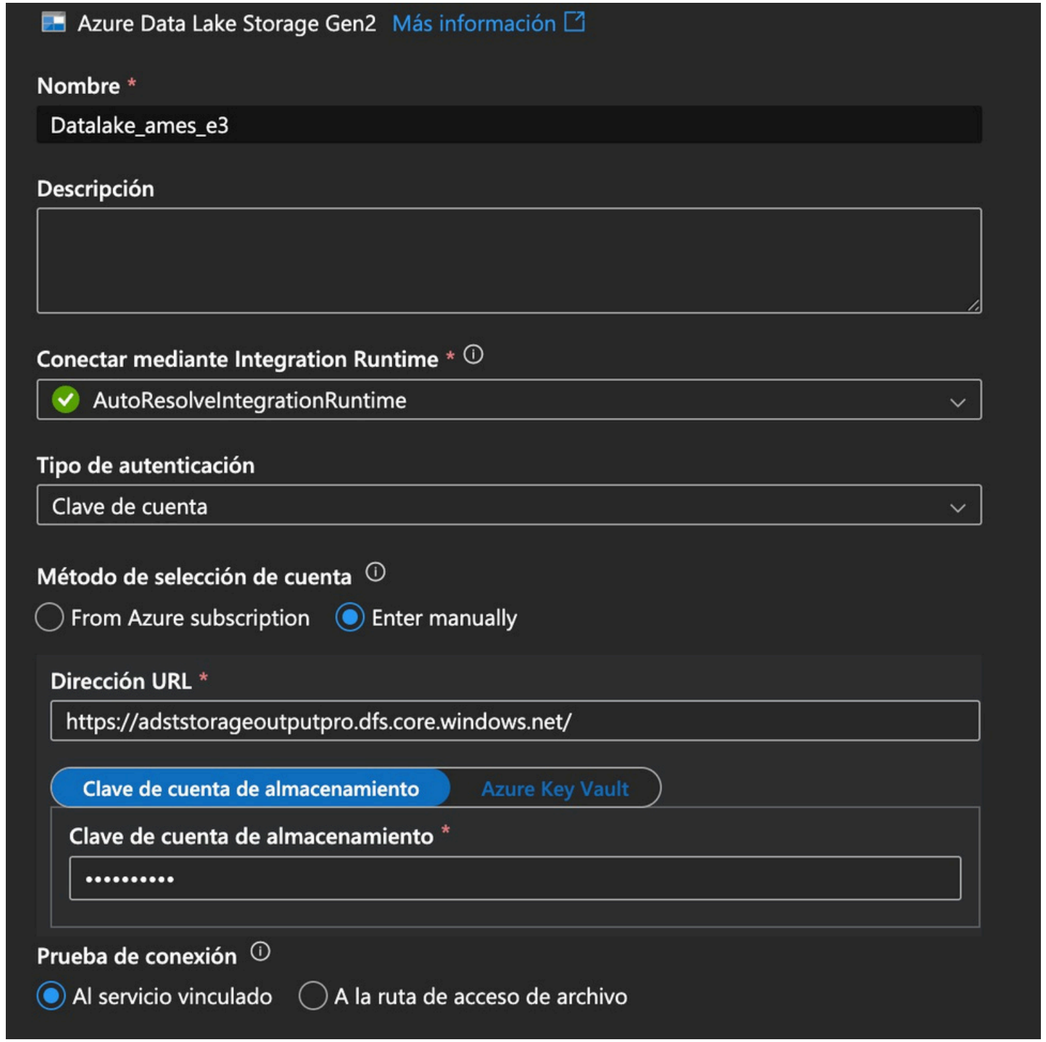
¿Por qué MongoDB Atlas para datos semi-estructurados?

✅ Flexibilidad para características opcionales
✅ Facilidad de consulta para documentos anidados
✅ Almacenamiento eficiente de datos variables

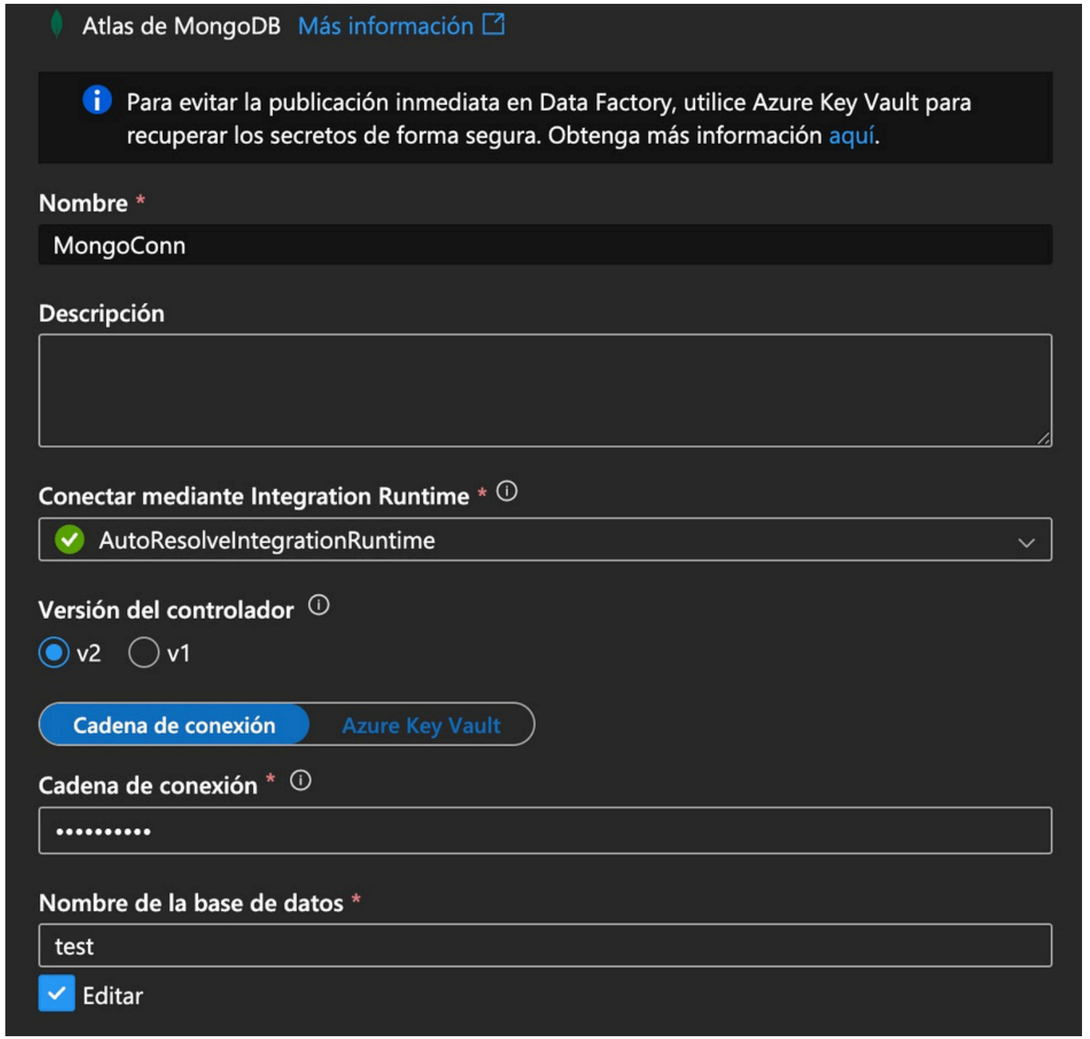
Estrategia de Unión:

- Clave primaria: PID (Property ID)
- Left Join para preservar todas las propiedades
- Manejo explícito de valores faltantes

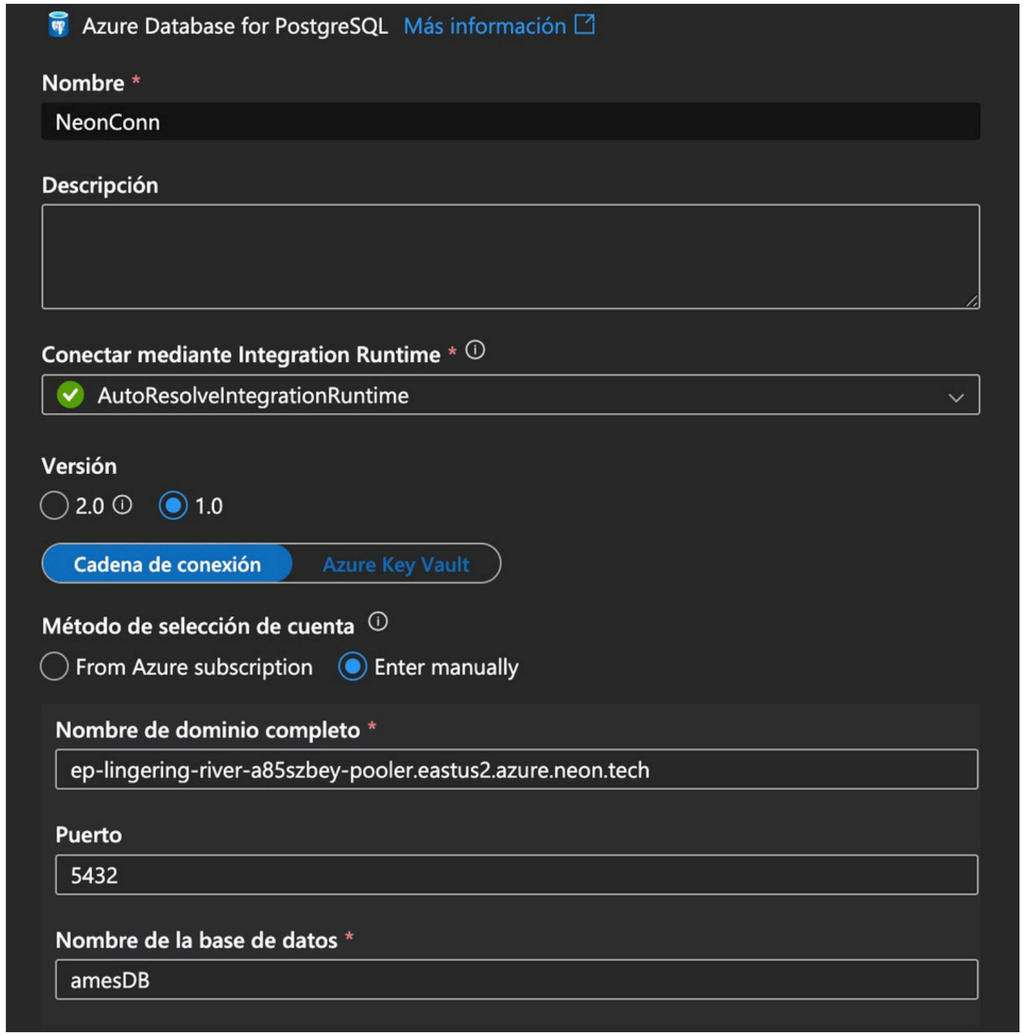
Configuración de Linked Services



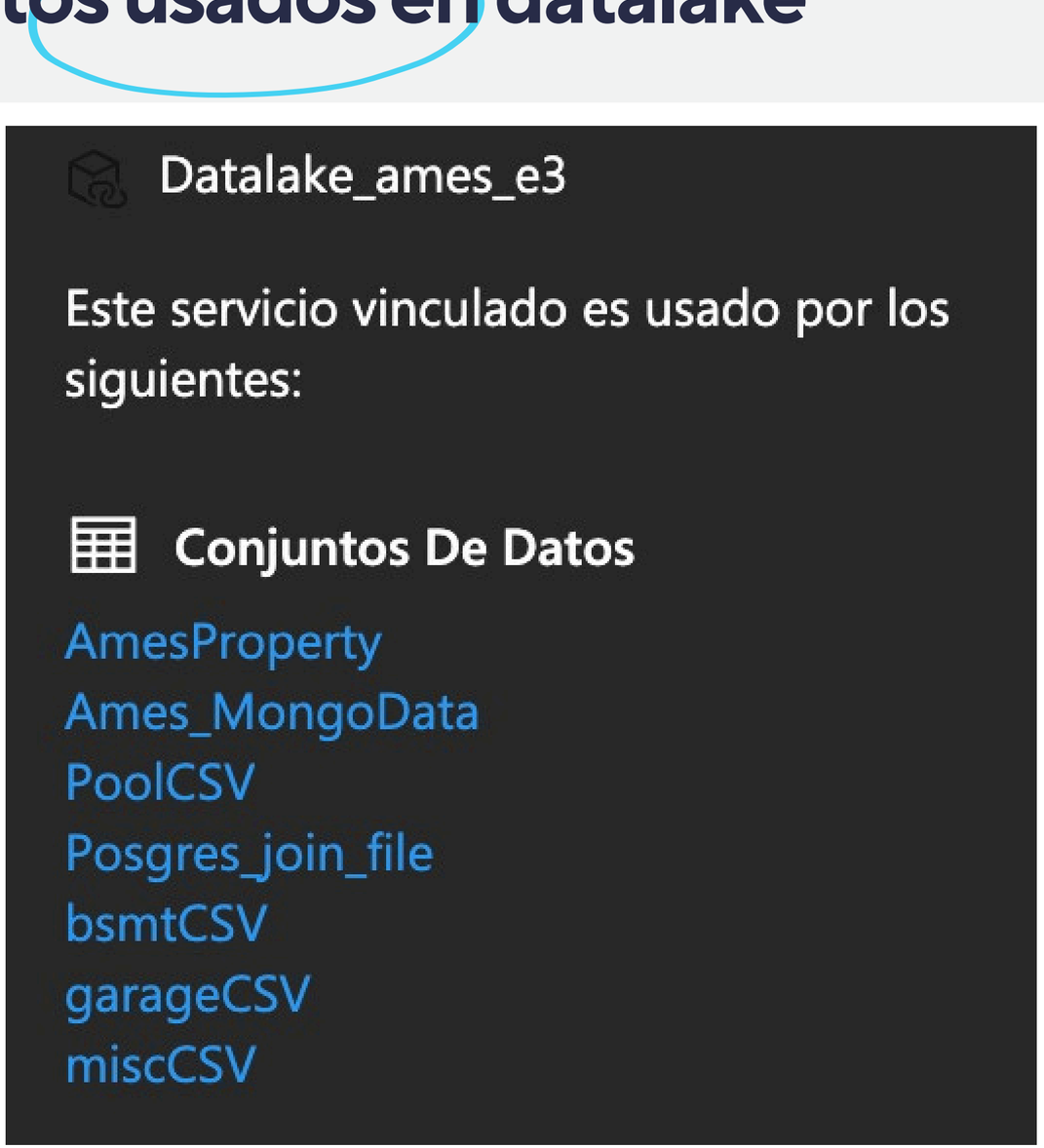
Configuración de Mongoconn



Configuración de Neonconn



Conjunto de datos usados en datalake



Transformaciones Implementadas - Parte 1

1. Cálculo de Campos Derivados
   - GrLivArea = 1stFlrSF + 2ndFlrSF
   - TotalBsmtSF = BsmtFinSF1 + BsmtFinSF2 + BsmtUnfSF
   - MoSold, YrSold: Extracción de fecha de venta

2. Agregaciones por Propiedad
   - Total Baños = FullBath + (HalfBath × 0.5) + BsmtFullBath + (BsmtHalfBath × 0.5)
   - TotRmsAbvGrd: Suma de habitaciones sobre nivel del suelo

3. Manejo de Valores Faltantes

   Regla aplicada:

   ├── Variables Cualitativas → "NA"

   └── Variables Cuantitativas → 0

Transformaciones Implementadas - Parte 2

Validaciones Implementadas:

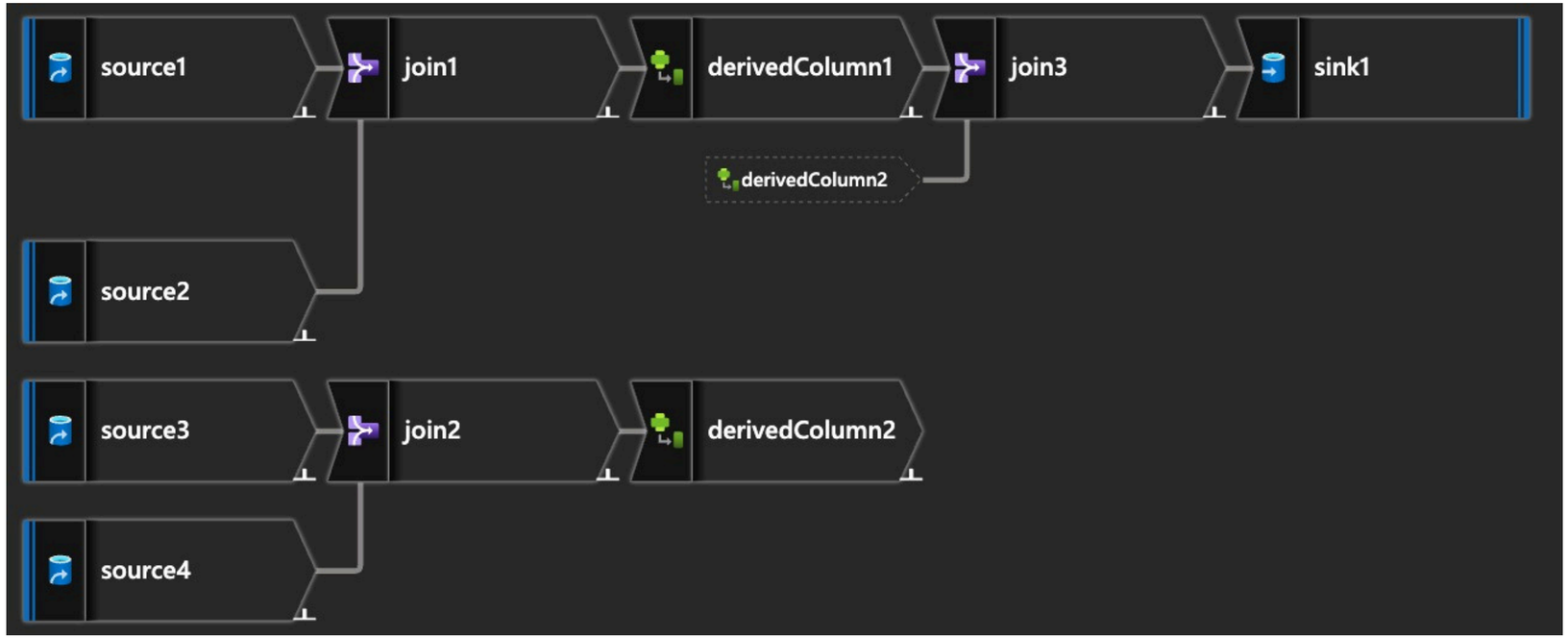
✓ Integridad Referencial
- Verificación de PID en todas las fuentes
- Validación de relaciones 1:N

✓ Rangos de Valores
- YearBuilt ≤ YrSold
- LotArea > 0
- SalePrice > 0

✓ Consistencia de Tipos
- Conversión de fechas a formato estándar
- Normalización de categorías (mayúsculas/minúsculas)

✓ Duplicados
- Eliminación de registros duplicados por PID

Data Flow - Captura Técnica



data flow donde se hace join de los mongos le pusimos Mongo_Join

Validación y Testing

Name	Last commit message	Last commit date
..		
Limpieza_y_tratamiento_E3_CTRL_...	Se añade notebook, consulta y salida	1 hour ago

Resultados - Archivo de Salida

Name	Last commit message	Last commit date
..		
out.md	Initial commit	2 weeks ago
salida.csv	Se añade notebook, consulta y salida	1 hour ago

Validación Automatizada:
Total tests: 20
Tests passed: 18
Percentage passed: 90.0%

[Incluir tu screenshot del código de validación]

Desafíos Encontrados

Integración de Múltiples Fuentes Problema:

Formatos y estructuras diferentes Solución:
• Mapeo detallado de campos (diseno_salida_campos.md)
• Data Flows para transformaciones complejas • Pruebas incrementales por fuente

Manejo de Datos Faltantes Problema:

Interpretación inconsistente de "faltante" Solución:
• Regla clara: NA para texto, 0 para números
• Documentación de cada caso
• Validación automatizada

Aprendizajes Técnicos

✓ Estrategias híbridas ETL/ELT
- Delegar transformaciones pesadas al motor SQL
- Mejora significativa de rendimiento

✓ Estandarización de tipos de datos
- Clave primaria (PID): consistencia es crítica
- Discrepancias enteros/strings causan pérdida de datos

✓ Complementariedad de herramientas
- ADF para orquestación
- Python para validación avanzada
- DBeaver para transformaciones SQL complejas

✓ Importancia de validación en capas
- Extracción → Transformación → Validación → Carga

Conclusiones

Estrategia Híbrida Optimizada

- Combinación única: SQL + ADF + Python
- Procesamiento distribuido según fortalezas de cada herramienta

Calidad Comprobada

- 90% de tests automatizados exitosos
- 0 valores nulos en dataset final
- 2,930 propiedades × 81 características

Arquitectura Replicable

- Documentación completa en GitHub
- Notebooks de validación disponibles
- Pipeline maestro versionado