



PIPELINE ETL PARA ANÁLISIS DE PROPIEDADES INMOBILIARIAS CON AZURE DATA FACTORY

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



CONTENIDO

1. Objetivo del Proyecto
2. Arquitectura del Pipeline ETL
3. Proceso de Extracción
4. Transformaciones Aplicadas
5. Resultados Obtenidos
6. Lecciones Aprendidas
7. Conclusiones y Mejoras Futuras



OBJETIVO



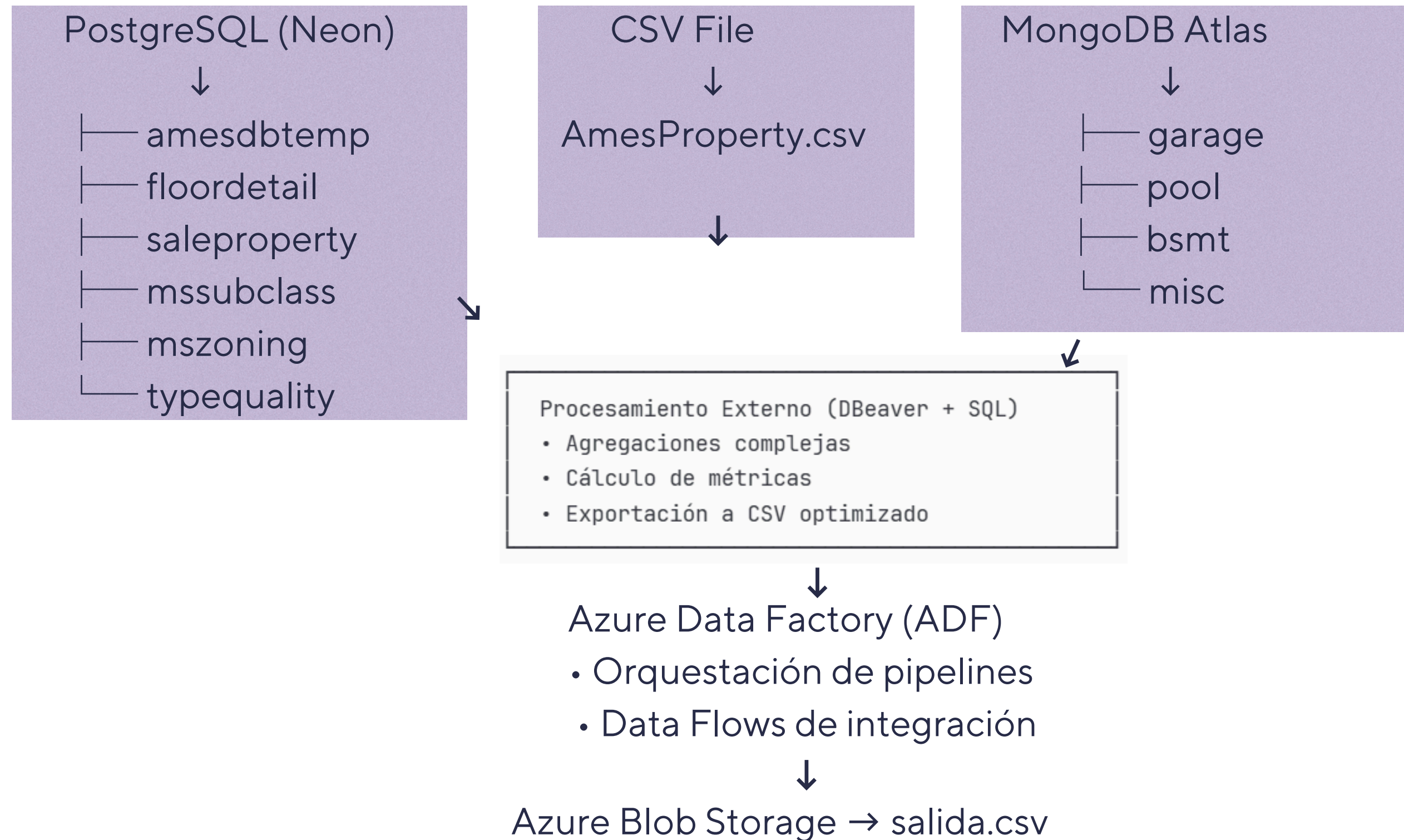
Diseñar e implementar un proceso ETL completo que integre datos de múltiples fuentes para generar un dataset unificado del Ames Housing Dataset.

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:
 - $\text{GrLivArea} = \text{1stFlrSF} + \text{2ndFlrSF} + \text{LowQualFinSF}$
 - 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



Azure Data Lake Storage Gen2 [Más información](#)

Nombre *
Datalake_ames_e3

Descripción

Conectar mediante Integration Runtime * ⓘ
☒ AutoResolveIntegrationRuntime

Tipo de autenticación
Clave de cuenta

Método de selección de cuenta ⓘ
☐ From Azure subscription ☒ Enter manually

Dirección URL *
https://adststorageoutputpro.dfs.core.windows.net/

Clave de cuenta de almacenamiento [Azure Key Vault](#)

Clave de cuenta de almacenamiento *
.....

Prueba de conexión ⓘ
☒ Al servicio vinculado ☐ A la ruta de acceso de archivo

Configuración de Mongoconn



Atlas de MongoDB [Más información](#)

i Para evitar la publicación inmediata en Data Factory, utilice Azure Key Vault para recuperar los secretos de forma segura. Obtenga más información [aquí](#).

Nombre *

MongoConn

Descripción

Conectar mediante Integration Runtime * ⓘ

✓ AutoResolveIntegrationRuntime

Versión del controlador ⓘ

☒ v2 ☐ v1

Cadena de conexión **Azure Key Vault**

Cadena de conexión * ⓘ

.....

Nombre de la base de datos *

test

☒ Editar

Configuración de Neonconn



 Azure Database for PostgreSQL [Más información](#) 

Nombre *

NeonConn

Descripción

Conectar mediante Integration Runtime * 

☒ AutoResolveIntegrationRuntime 

Versión

☐ 2.0  ☒ 1.0

Cadena de conexión

Azure Key Vault

Método de selección de cuenta 

☐ From Azure subscription ☒ Enter manually

Nombre de dominio completo *

ep-lingering-river-a85szbey-pooler.eastus2.azure.neon.tech

Puerto

5432

Nombre de la base de datos *

amesDB

Conjunto de datos usados en datalake



Datalake_ames_e3

Este servicio vinculado es usado por los siguientes:



Conjuntos De Datos

AmesProperty

Ames_MongoData

PoolCSV

Posgres_join_file

bsmtCSV

garageCSV

miscCSV

Transformaciones Implementadas - Parte 1



1. Cálculo de Campos Derivados

- $\text{GrLivArea} = \text{1stFlrSF} + \text{2ndFlrSF}$
- $\text{TotalBsmtSF} = \text{BsmtFinSF1} + \text{BsmtFinSF2} + \text{BsmtUnfSF}$
- MoSold, YrSold: Extracción de fecha de venta

2. Agregaciones por Propiedad

- $\text{Total Baños} = \text{FullBath} + (\text{HalfBath} \times 0.5) + \text{BsmtFullBath} + (\text{BsmtHalfBath} \times 0.5)$
- 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

- $\text{YearBuilt} \leq \text{YrSold}$
- $\text{LotArea} > 0$
- $\text{SalePrice} > 0$

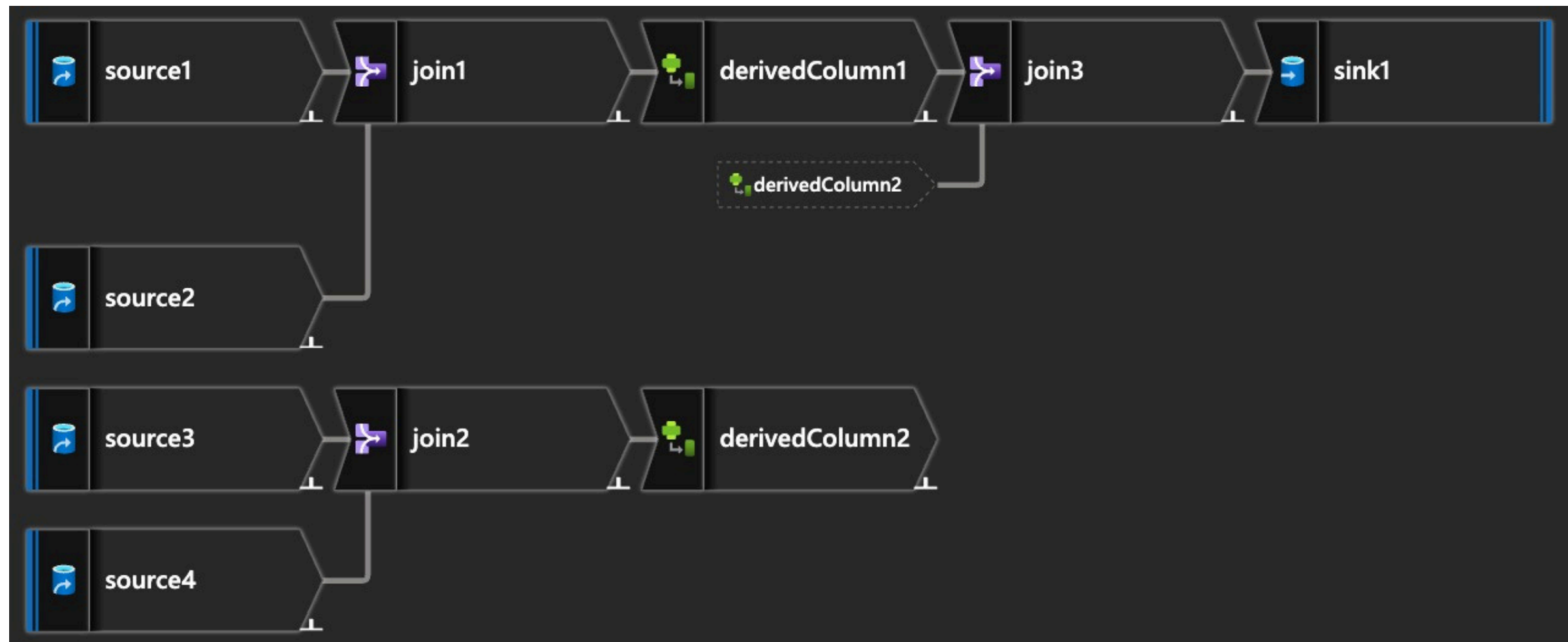
✓ Consistencia de Tipos

- 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