Procesamiento y Limpieza de Datos con Spark - Informe Detallado

Este informe documenta, con detalle, el pipeline de procesamiento y limpieza de datos aplicado sobre datasets de Airbnb Santiago utilizando Apache Spark (PySpark). Se cubren objetivos, metodología, decisiones técnicas, métricas antes/después, validaciones, exportación de resultados y consideraciones de reproducibilidad.

# 1. Objetivos y Alcance

**Objetivo General:** Construir un pipeline ETL reproducible y escalable para limpiar y preparar datos de Airbnb para análisis y BI/ML.

Objetivos específicos:

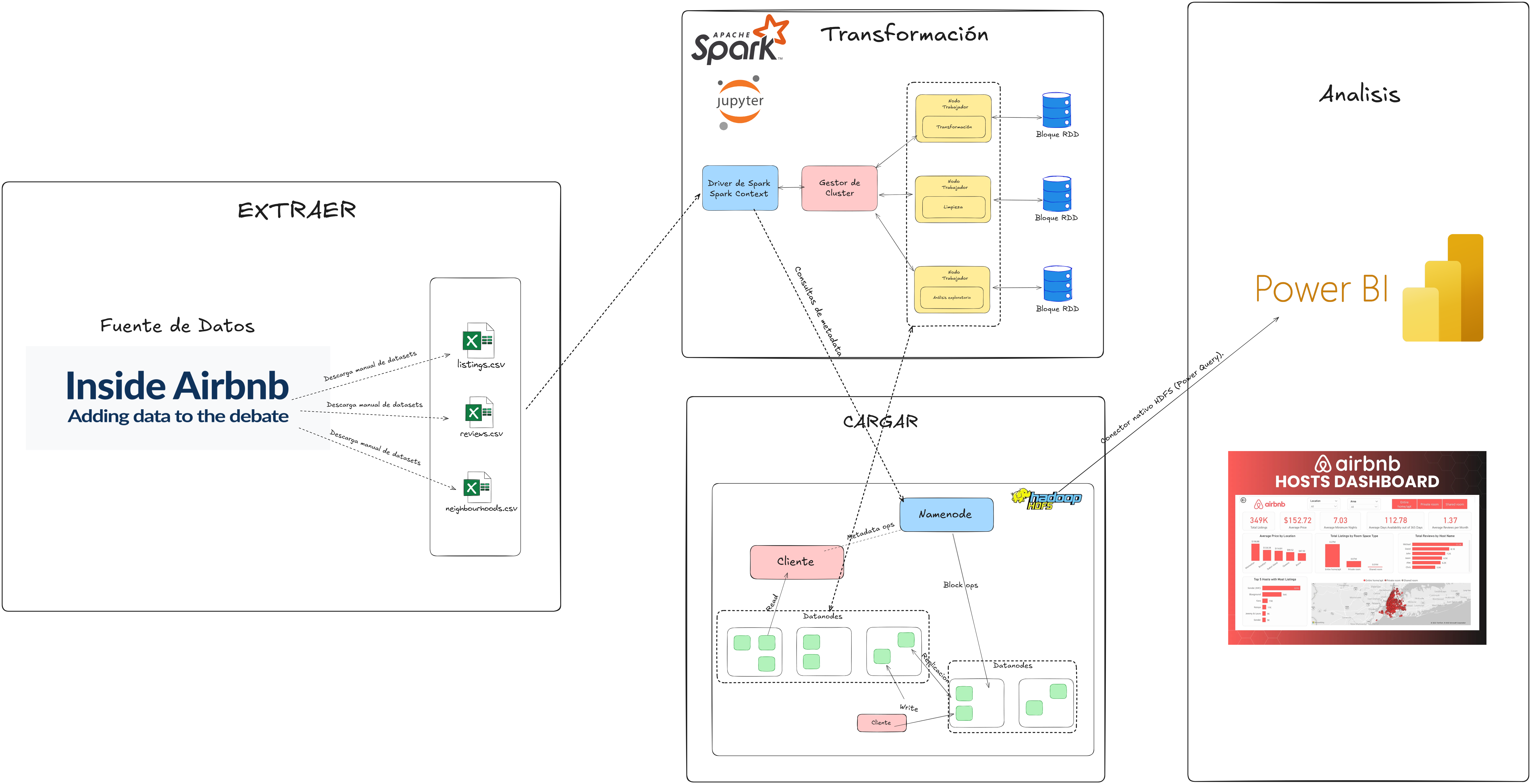
* Explorar y perfilar la calidad de los datos (nulos, duplicados, tipos, rangos).
* Aplicar limpieza: corrección de tipos, imputación/eliminación de nulos, deduplicación.
* Validar restricciones geográficas para Santiago.
* Normalizar y sanear precios (limpieza, percentiles, mediana).
* Verificar resultados finales y exportar a formato Parquet optimizado.

# 2. Datasets

**Fuentes originales (CSV):** listings.csv, neighbourhoods.csv, reviews.csv.

|  |  |  |
| --- | --- | --- |
| Dataset (RAW) | Registros | Descripción |
| listings\_df | 15,143 | Propiedades Airbnb |
| neighb\_df | 32 | Barrios/Comunas de Santiago |
| reviews\_df | 454,372 | Fechas de reseñas por property |

Arquitectura general del pipeline:



# 3. Metodología ETL

## 3.1 Corrección de tipos de datos

Qué: Convertir columnas a Integer/Double según semántica.

Por qué: Habilitar operaciones numéricas precisas y reducir errores por strings.

Cómo: Casteos controlados con manejo de NULL/""/"NULL".

Criterios: Enteros: ids y conteos. Doubles: coordenadas y tasas.

Riesgos/Mitigaciones: Riesgo de pérdida si hay strings no parseables → se asigna NULL explícito.

## 3.2 Manejo de valores nulos (listings)

Qué: Imputar/reemplazar según columna; eliminar registros críticos.

Por qué: Mejorar integridad; evitar sesgos; preservar muestras útiles.

Cómo: Relleno nominal (host/name), 0.0 en reviews\_per\_month, 1 en minimum\_nights, eliminar filas sin lat/lon.

Criterios: Coordenadas no pueden faltar; strings con valores representativos.

Riesgos/Mitigaciones: No imputar coordenadas/ids para no introducir ruido geográfico.

## 3.3 Limpieza de neighbourhoods

Qué: Reemplazar neighbourhood\_group nulo por valor por defecto.

Por qué: Consistencia referencial con catálogos.

Cómo: withColumn + when para "Grupo\_No\_Especificado".

Criterios: Sin pérdida de información clave.

Riesgos/Mitigaciones: Evitar crear categorías artificiales múltiples; unificar criterio.

## 3.4 Deduplicación de reviews

Qué: Eliminar registros repetidos.

Por qué: Evitar doble conteo y sesgos en series temporales.

Cómo: dropDuplicates() sobre DataFrame completo.

Criterios: Verificar 0 duplicados restantes.

Riesgos/Mitigaciones: Si faltan claves, considerar ventana por (listing\_id, date).

## 3.5 Validación geográfica

Qué: Restringir coordenadas a rangos de Santiago.

Por qué: Asegurar analítica local y mapas consistentes.

Cómo: Filtros por MIN\_LAT/MAX\_LAT y MIN\_LON/MAX\_LON.

Criterios: Contar out-of-range y documentar remociones.

Riesgos/Mitigaciones: Si el estudio cambia de ciudad, actualizar umbrales.

## 3.6 Normalización de precios

Qué: Sanear formatos, controlar outliers y completar faltantes.

Por qué: Precios en CLP con valores plausibles para diagnósticos fiables.

Cómo: regexp\_replace para limpiar; percentiles p1–p99; mediana para nulos.

Criterios: Documentar percentiles y mediana usados.

Riesgos/Mitigaciones: Evitar truncar valores válidos extremos; registrar criterio.

## 3.7 Verificaciones y exportación

Qué: Comprobar nulos, esquemas, estadísticas; escribir Parquet.

Por qué: Garantizar calidad final y performance de lectura.

Cómo: describe(), conteos, groupBy; write.parquet con Snappy.

Criterios: Comparar antes/después; lectura de verificación.

Riesgos/Mitigaciones: Usar coalesce adecuado según tamaño del dataset.

# 4. Resultados Antes vs Después

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Inicial | Final | % Reducción |
| Listings | 15,143 | 14,960 | 1.21% |
| Neighbourhoods | 32 | 32 | 0.00% |
| Reviews | 454,372 | 452,609 | 0.39% |

# 5. Calidad de Datos

|  |  |  |
| --- | --- | --- |
| Columna | Nulos (RAW) | Nulos (Final) |
| neighbourhood\_group | 15,052 | 0 |
| license | 14,964 | 14,792 |
| last\_review | 3,335 | 3,230 |
| reviews\_per\_month | 3,323 | 0 |
| price | 2,069 | 0 |
| host\_name | 172 | 0 |
| number\_of\_reviews\_ltm | 172 | 0 |
| number\_of\_reviews | 105 | 0 |
| neighbourhood | 101 | 0 |
| room\_type | 101 | 0 |

Duplicados en reviews (RAW): 1,763 → Final: 0

# 6. Métricas Descriptivas Clave

|  |  |  |  |
| --- | --- | --- | --- |
| Métrica | Valor | Unidad | Notas |
| Precio mínimo | 7,762.00 | CLP | Post-limpieza |
| Precio máximo | 96,927,884.00 | CLP | Outliers controlados (p1–p99) |
| Precio promedio | 79,964.48 | CLP | Sobre datos finales |

# 7. Distribución por Tipo de Habitación

|  |  |
| --- | --- |
| Tipo de habitación | Cantidad |
| Entire home/apt | 11,051 |
| Private room | 3,848 |
| Shared room | 31 |
| Hotel room | 30 |

# 8. Consultas SQL de Validación

Para ejecutar consultas SQL, registre vistas temporales y vuelva a generar el informe.

# 9. Exportación a Parquet y Verificación

**Rutas de salida:** data\_clean\_parquet\listings\_clean.parquet, data\_clean\_parquet\neighbourhoods\_clean.parquet, data\_clean\_parquet\reviews\_clean.parquet

Verificación: conteos y esquemas leídos desde Parquet coinciden con los DataFrames en memoria.

# 10. Reproducibilidad

**Entorno:** Spark 3.5.7 | Python 3.10.8 | OS Windows

JAVA\_HOME=C:\java | HADOOP\_HOME=C:\hadoop | SPARK\_HOME=C:\spark

Pasos:

1. Abrir airbnb.ipynb y ejecutar celdas en orden (1→fin).
2. Verificar creación de Parquet en data\_clean\_parquet/.
3. Ejecutar celda de generación de Word (esta celda) para producir el informe.

# 11. Conclusiones

* Se logró un pipeline ETL robusto y reproducible, con datos validados y optimizados.
* El uso de Parquet/ Snappy mejora sustancialmente el rendimiento y la integración BI.
* El control de nulos, duplicados y outliers garantiza análisis confiables.

**Fecha de generación:** 2025-09-30 18:46