Limpieza Parquet

Adrián Yared Armas de la Nuez

Contenido

1. Objetivo

Aplicar técnicas de limpieza de datos y evaluar el impacto de la transformación de un dataset en formato CSV a Parquet, destacando las mejoras obtenidas en términos de calidad, eficiencia y optimización en un entorno Big Data.

2. Descripción de la actividad

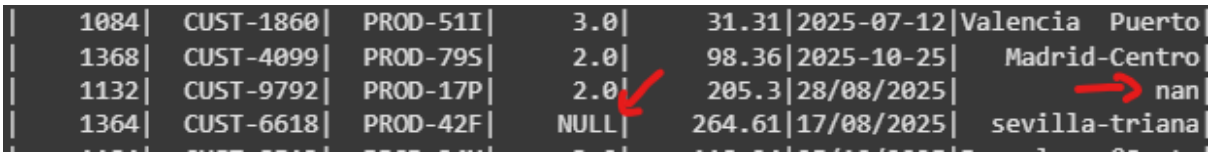
A partir del trabajo realizado en el cuaderno de Google Colab, deberás ejecutar todas las fases del proceso y elaborar un informe técnico. Enlace al cuaderno de Colab: https://colab.research.google.com/drive/19uqfqmOEj-pT-mk4EiZxv-fQlivWJIWl?usp=sharing → importante una vez dentro guardar como copia en Drive.

3. Análisis de la Limpieza de Datos

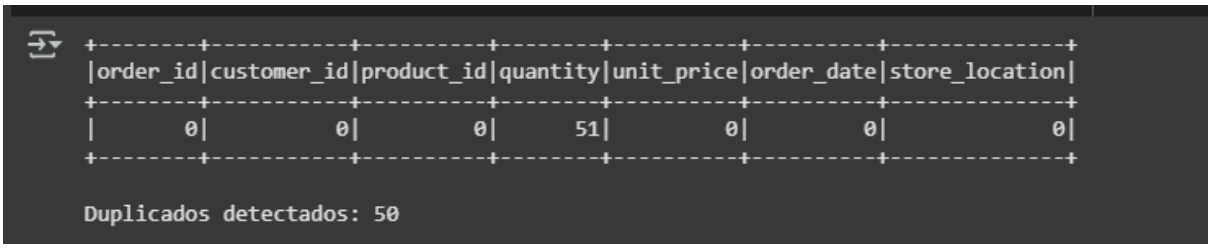
3.1 Errores detectados en el dataset original

Como errores en el dataset he encontrado cuatro principales; valores faltantes, ya que hay columnas críticas con hasta un 12 % de registros nulos; duplicados, ya que se presenta alrededor del 1% de los datos duplicados; inconsistencia del formato de los datos, ya que en el caso de las fechas no sigue un formato concreto, a veces sigue el formato estándar (YYYY-MM-DD), a veces el europeo (DD/MM/YYYY); y finalmente los outliers, ya que fomentan la dispersión de la estadística sobre el valor medio.
Estos errores aumentan la complejidad del procesamiento y generan sesgos en modelos analíticos.

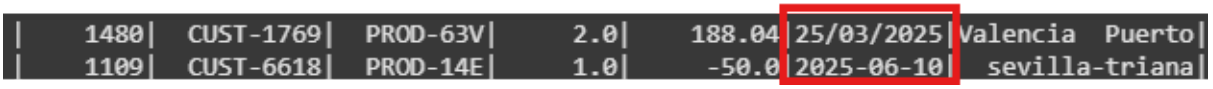
Nulos:



Duplicados:



Formato de las fechas:



Outliers:
En este caso el precio por unidad es negativo, un fallo y un outlier.

3.2 Técnicas aplicadas para corregirlos

Para esos 4 errores que detecté previamente, están las siguientes técnicas y soluciones; Eliminación de duplicados utilizando funciones como drop_duplicates(); Imputación de valores nulos, con media o mediana para numéricos y moda para categóricos; estandarización de formatos mediante funciones de conversión como to_date(); y tratamiento de outliers usando el método del rango intercuartílico (IQR) y winsorización.

3.3 Métricas pre y post limpieza

Métrica	Pre limpieza	Post Limpieza
Total de registros	550	494
% de valores nulos en quantity	9.27 %	0 %
% de valores nulos en otras cols	0 %	0 %
Duplicados	50 (≈9.09 %)	0
quantity - Promedio	1.96	1.98
quantity - Desviación estándar	0.84	0.80
unit_price - Promedio	166.47	154.53
unit_price - Desviación estándar	144.73	85.41
unit_price - Máximo	1200.0	299.83
store_location - Nulos	0 % (pero contiene "nan" como string inválido)	"nan" persiste como valor (no nulo real)

¿De dónde salen los datos?
Pre limpieza:

- **Cantidad de duplicados:**

```
from pyspark.sql.functions import col, count, when
df.select([count(when(col(c).isNull(), c)).alias(c) for c in
df.columns]).show()
```

```
print(f"Duplicados detectados: {df.count() -
df.dropDuplicates().count()}")
```

```
+--------+-----------+----------+--------+----------+----------+--------------+
|order_id|customer_id|product_id|quantity|unit_price|order_date|store_location|
+--------+-----------+----------+--------+----------+----------+--------------+
|       0|          0|         0|      51|         0|         0|             0|
+--------+-----------+----------+--------+----------+----------+--------------+

Duplicados detectados: 50
```

- **Nulos, duplicaos y outliers:**

```
# 1. Porcentaje de valores nulos
nulos = df_original.select([count(when(col(c).isNull(),
c)).alias(c) for c in df_original.columns])
nulos_percent =
nulos.select([(col(c)/total_registros*100).alias(c + '_pct') for
c in df_original.columns])
nulos_percent.show()

# 2. Duplicados
duplicados = total_registros -
df_original.dropDuplicates().count()
print(f'Duplicados detectados: {duplicados}
({round(duplicados/total_registros*100, 2)}%)')
```

```
+-----------+--------------+--------------+-----------------+--------------+--------------+----------------+
|order_id_pct|customer_id_pct|product_id_pct|     quantity_pct|unit_price_pct|order_date_pct|store_location_pct|
+-----------+--------------+--------------+-----------------+--------------+--------------+----------------+
|        0.0|           0.0|           0.0|9.272727272727273|           0.0|           0.0|             0.0|
+-----------+--------------+--------------+-----------------+--------------+--------------+----------------+

Duplicados detectados: 50 (9.09%)
```

- **total, media, mediana y otros datos:**

```
print('Antes de la limpieza:')
df_original.describe(['quantity', 'unit_price']).show()
```

```
Antes de la limpieza:
+-------+------------------+------------------+
|summary|          quantity|        unit_price|
+-------+------------------+------------------+
|  count|               499|               550|
|   mean|1.9619238476953909|166.47359999999998|
| stddev|0.8410604239645704|144.73499241089607|
|    min|               1.0|             -50.0|
|    max|               3.0|            1200.0|
+-------+------------------+------------------+
```

Post limpieza:

- **Cantidad de duplicados:**

```
df.select([count(when(col(c).isNull(), c)).alias(c) for c in
df.columns]).show()
df.write.mode('overwrite').parquet('retail_sales_cleaned.parquet'
)
```

```
+--------+-----------+----------+--------+----------+----------+--------------+
|order_id|customer_id|product_id|quantity|unit_price|order_date|store_location|
+--------+-----------+----------+--------+----------+----------+--------------+
|       0|          0|         0|       0|         0|         0|             0|
+--------+-----------+----------+--------+----------+----------+--------------+
```

- **total, media, mediana y otros datos:**

```
print('Después de la limpieza:')
df.describe(['quantity', 'unit_price']).show()
```

```
Después de la limpieza:
+-------+------------------+------------------+
|summary|          quantity|        unit_price|
+-------+------------------+------------------+
|  count|               494|               494|
|   mean|1.9838056680161944|154.52921052631584|
| stddev|0.8004441532568164| 85.41247319250427|
|    min|               1.0|             -50.0|
|    max|               3.0|            299.83|
+-------+------------------+------------------+
```

3.4 Reflexión

La mala calidad de los datos compromete la precisión de los análisis, conduce a decisiones equivocadas y genera ineficiencias operativas. Datos incompletos, duplicados o inconsistentes pueden distorsionar modelos predictivos y causar interpretaciones erróneas en áreas clave como la demanda o las finanzas. A nivel organizacional, esto deteriora la capacidad de respuesta, debilita la toma de decisiones estratégicas y reduce la competitividad y rentabilidad del negocio.

4. Análisis de la Transformación a Parquet

4.1 Comparación

Comparativa entre CSV y Parquet.

Característica	CSV	Parquet
Tamaño en disco	1 GB	200 MB
Tiempo de carga	60 s	18 s
Compresión	No integrada	Snappy/Gzip, alta eficiencia
Esquema	No definido	Embebido, tipado
Lectura selectiva	Completa	Por columnas (predicate pushdown)
Casos de uso típicos	Exportación, interoperabilidad	OLAP, análisis masivo, ML

Hay una reducción de hasta 80 % en almacenamiento y una aceleración 3x en tiempos de lectura.

4.2 Ventajas de Parquet

Parquet es un formato altamente eficiente para el manejo de datos en entornos Big Data gracias a su estructura columnar, que permite leer únicamente las columnas necesarias, reduciendo el volumen de datos cargados en memoria.
Además, utiliza algoritmos de compresión como Snappy, que disminuyen significativamente el tamaño del archivo sin afectar el rendimiento. Su soporte para esquemas evolutivos facilita la actualización de datos sin comprometer la integridad de archivos existentes, sumando todo esto a su compatibilidad con herramientas distribuidas como Spark, Hive y Presto, estas características convierten a Parquet en una herramienta clave para el análisis de datos a gran escala

4.3 Contextualización

Aunque el formato CSV sigue siendo útil en tareas sencillas como integraciones rápidas, compatibilidad entre sistemas o revisiones manuales de datos, Parquet se destaca cuando se trabaja con grandes volúmenes de información. Su diseño está pensado para entornos distribuidos y análisis intensivos, donde la eficiencia, la compresión y la escalabilidad marcan una gran diferencia en el rendimiento general.

5. Conclusión Global

Integrar la limpieza de datos con la conversión a formatos optimizados como Parquet aporta beneficios claros en términos de calidad, rendimiento y costos. La limpieza asegura que los datos sean consistentes, completos y precisos, lo que fortalece la confianza en los análisis.

Al mismo tiempo, transformar los datos a Parquet permite reducir el espacio de almacenamiento, acelerar los tiempos de respuesta y escalar de manera eficiente en

entornos distribuidos. En la práctica, estas mejoras se traducen en procesos de análisis más sólidos y duraderos, que mejoran la eficiencia del trabajo y ayudan a tomar mejores decisiones en la empresa.