Empresas Carozzi

Desafío Data Engineer

Autor: Sebastián Garcés

Diciembre - 2025

Contexto del Desafío

La compañía vende a clientes del canal tradicional (almacenes, botillerías, kioscos, etc.) a través de un canal de ventas transaccional. El equipo de Analítica Avanzada necesita construir un modelo de segmentación de clientes y necesita una vista consolidada a nivel cliente, con características (features) calculadas a partir de las transacciones históricas solicitada por los data scienstist del área para generar este modelo de segmentación de clientes.

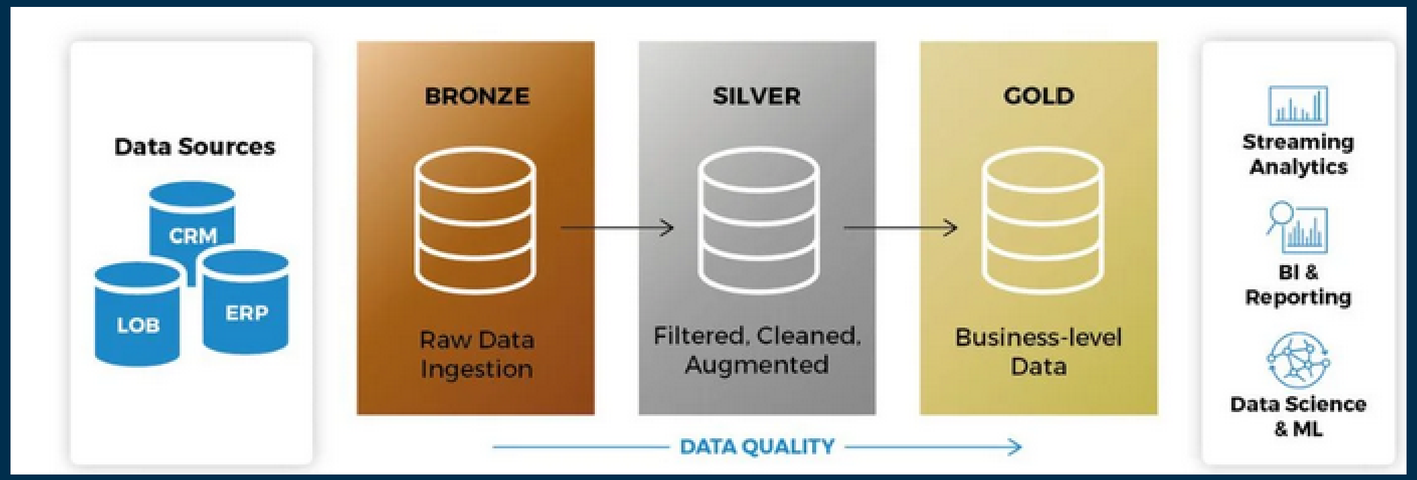
Objetivo del Desafío

Diseñar e implementar una arquitectura tipo Medallion (Bronze / Silver / Gold) usando PySpark, ejecutable en tu computador local, que genere en la capa Gold la siguiente tabla:

- **dim_features**: tabla de características de clientes calculadas de los últimos tres meses de datos disponibles en los datos transaccionales de ventas

Además, deberás documentar y presentar cómo llevarías esta solución a un entorno de Azure / Microsoft Fabric.

Arquitectura General Propuesta



Ventajas de Medallón:

- Mejorar la calidad del dato de forma incremental.
- Simplificar la trazabilidad del pipeline.
- Crear datasets reutilizables, consistentes y confiables.
- Separar responsabilidades de ingesta, limpieza y consumo analítico.
- Facilitar auditoría, reprocesamientos y escalabilidad.

Bronze:

- Almacenar datos tal como llegan.
- Orientado a la optimizacion de almacenamiento.
- Mantener un *single source of truth*.
- Soporte a auditoría y reprocesos.

Silver:

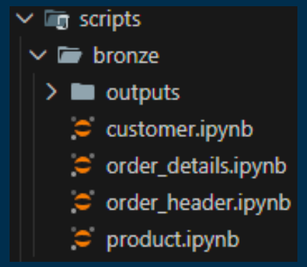
- Refinar datos desde Bronze.
- Aplicar reglas de negocio, limpieza y estandarización.
- Se define el lenguaje comun entre negocio, analitica y TI.
- Construir tablas dimensionles y de hechos.

Gold:

- Datasets curados y diseñados para entregar valor al negocio: reportes, dashboards, analítica avanzada y machine learning
- Se crean KPI, metricas compuestas, agregaciones, cohorters, periodos moviles, features de ML, etc.

Solución Implementada

Bronze



- Se generan 1 Notebook de carga por tabla a ingestar , facilitando la posterior operación y mantenibilidad del codigo.

- En esta fase se agregan el timestamp de ingestion y el archivo fuente a datos.

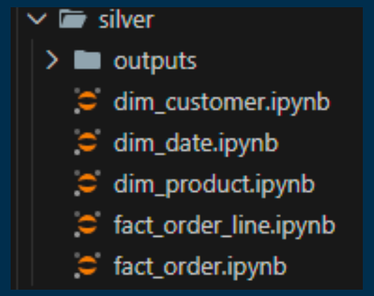
```
# Agregar columnas de control
df_bronze = (
    df_raw
    .withColumn("ingestion_timestamp",   current_timestamp())       # timestamp exacto
    .withColumn("source_file",    input_file_name())         # ruta del archivo origen
)
```

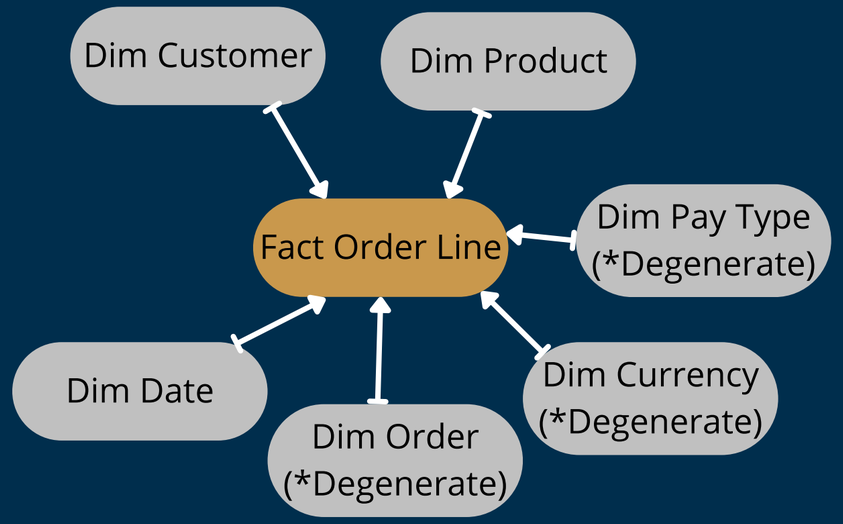
En el caso de origenes de datos framentados se utliza metodo de busqueda por nombres usando wild cards, luego Pyspark se encarga de unificar la data disperza.

```
# Path con wildcard para lecturas masivas
search_pattern = raw_path + "/**/*orders_detail*.csv" # patron para buscar todos los archivos con nombre order_header

#Buscar archivos que coincidan con el patrón
files = glob.glob(search_pattern, recursive=True)

if len(files)    0:
```

Solución Implementada

Silver





* No tiene tabla propia. Atributo único y textual dentro de la fact.

- 1 Notebook por Dimension/ Hecho a Ingestar.
- Se crea Dim Date para no perder referencias de fechas de transacciones.
- Ademas se crea como "Bonus" fact_order para futuros analisis

Se aplican las siguientes transformaciones y limpiezas:

1. Deddup mediante funciones de Ventanas por id de tabla. ROW_NUMBER() OVER(PARTITION BY...)
2. Transformaciones a tipo de datos correctos.
3. En tablas "maestros" se agregan datos Dummy.
4. Posibilidad de SCD para manejo de historial de cambios en dato (mejora en Fabric/Delta Tables)
5. Filtrado de datos a solicitud del negocio en tablas Fact. head.order_net_amount > 0 OR line.qty_units > 0

Dim Customer

```
customer_id: integer (nullable = false)
customer_name: string (nullable = true)
channel: string (nullable = true)
actual_segment: string (nullable = true)
region: string (nullable = true)
comuna: string (nullable = true)
entry_date: timestamp (nullable = true)
status: string (nullable = true)
mix_credito_vs_contado_3m: double (nullable = true)
```

customer_id	customer_name	channel	actual_segment	region	comuna	entry_date	status	mix_credito_vs_contado_3m
0	Sin Cliente	Sin Canal	N/A	N/A	N/A	NULL	NULL	NULL
1	Cliente_00001	otros	B	Región Metropolitana	Maipú	2022-04-16 10:00:...	activo	0.0
2	Cliente_00002	otros	A	Biobío	Concepción	2022-02-21 10:00:...	activo	0.0
3	Cliente_00003	tradicional	C	Biobío	Concepción	2024-06-19 10:00:...	activo	0.28177114666661934
4	Cliente_00004	tradicional	C	Biobío	Talcahuano	2023-06-18 10:00:...	inactivo	0.4796695637737919
5	Cliente_00005	tradicional	B	Antofagasta	Antofagasta	2023-03-05 10:00:...	activo	0.6322392118803829
6	Cliente_00006	tradicional	B	Biobío	Talcahuano	2023-04-03 10:00:...	activo	0.45105535032968525
7	Cliente_00007	otros	D	Maule	Talca	2022-04-21 10:00:...	activo	0.25114183544864904
8	Cliente_00008	tradicional	C	Valparaíso	Viña del Mar	2021-04-11 10:00:...	activo	0.41163295216238105
9	Cliente_00009	tradicional	C	Valparaíso	Valparaíso	2025-08-31 10:00:...	activo	0.7031062877390177
10	Cliente_00010	tradicional	B	Maule	Curicó	2024-08-10 10:00:...	activo	0.5557985871093494
11	Cliente_00011	tradicional	B	Región Metropolitana	Santiago	2021-11-16 10:00:...	activo	0.2903640469224722
12	Cliente_00012	tradicional	D	Región Metropolitana	Maipú	2023-12-12 10:00:...	activo	0.18426067328930873
13	Cliente_00013	tradicional	C	Región Metropolitana	La Florida	2023-02-26 10:00:...	activo	0.7914977655006172
14	Cliente_00014	tradicional	B	Región Metropolitana	Santiago	2021-04-05 10:00:...	activo	0.3523982270780352
15	Cliente_00015	tradicional	C	Antofagasta	Calama	2024-04-21 10:00:...	activo	0.0
16	Cliente_00016	otros	C	Maule	Talca	2023-03-15 10:00:...	activo	0.11236617264411805
17	Cliente_00017	tradicional	B	Valparaíso	Viña del Mar	2025-09-16 10:00:...	activo	0.2944330276347585
18	Cliente_00018	tradicional	B	Maule	Curicó	2025-10-09 10:00:...	activo	0.11961348658734408
19	Cliente_00019	otros	B	Valparaíso	Valparaíso	2025-05-12 10:00:...	activo	0.0

showing top 20 rows

Fact Order Line

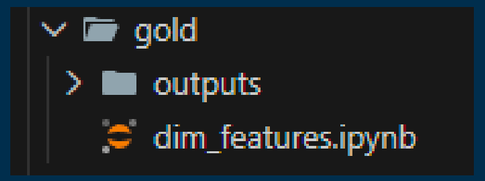
```
root
 |-- order_line_key: string (nullable = false)
 |-- order_id: integer (nullable = false)
 |-- line_id: integer (nullable = false)
 |-- product_id: integer (nullable = true)
 |-- customer_id: integer (nullable = true)
 |-- order_date_id: integer (nullable = true)
 |-- period: integer (nullable = true)
 |-- channel: string (nullable = true)
 |-- pay_type: string (nullable = true)
 |-- currency: string (nullable = true)
 |-- promo_flag: integer (nullable = true)
 |-- qty_units: integer (nullable = true)
 |-- unit_price: decimal(18,2) (nullable = true)
 |-- line_net_amount: decimal(18,2) (nullable = true)
 |-- order_net_amount_replicated: decimal(18,2) (nullable = true)


11241634
```

Rows: 11.241.634

Partition: Periodos

Solución Implementada

Gold



Todas las reglas de negocios fueron aplicadas mediante SparkSQL

Grano minino: Cliente (customer_id)
Periodo de tiempo de analisis: Ultimos 3 Meses

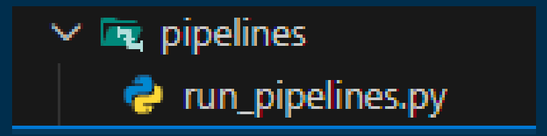
```
root
 |-- customer_id: integer (nullable = true)
 |-- recencia_dias: integer (nullable = true)
 |-- ventas_total_3m: decimal(28,2) (nullable = true)
 |-- frecuencia_pedidos_3m: long (nullable = false)
 |-- ticket_promedio_3m: decimal(38,12) (nullable = true)
 |-- dias_promedio_entre_pedidos: double (nullable = true)
 |-- variedad_categorias_3m: long (nullable = false)
 |-- porcentaje_pedidos_promo_3m: double (nullable = true)
 |-- mix_credito_vs_contado_3m: decimal(38,10) (nullable = true)

+-----------+-------------+---------------+----------------------+--------------------+---------------------------+----------------------+---------------------------+-------------------------+
|customer_id|recencia_dias|ventas_total_3m|frecuencia_pedidos_3m| ticket_promedio_3m|dias_promedio_entre_pedidos|variedad_categorias_3m|porcentaje_pedidos_promo_3m|mix_credito_vs_contado_3m|
+-----------+-------------+---------------+----------------------+--------------------+---------------------------+----------------------+---------------------------+-------------------------+
|       8638|           32|     5856112.71|                    40|146402.817750000000|          1.4615384615384615|                     5|                      0.925|             0.3378087646|
|       2366|           49|     1415444.08|                     8|176930.510000000000|          4.142857142857143|                     5|                        1.0|             0.2288744604|
|       1829|           32|     8510613.80|                    56|151975.246428571429|          1.0727272727272728|                     5|        0.9107142857142857|             0.0207661497|
|       9427|           30|     4384557.95|                    29|151191.653448275862|          2.1785714285714284|                     5|        0.9655172413793104|             0.9308037131|
|       7253|           43|     1226355.57|                     6|204392.595000000000|                        6.4|                     5|                        1.0|             0.3924426991|
|        496|           44|     1824740.59|                    11|165885.508181818182|                        4.3|                     5|        0.8181818181818182|             0.1881318265|
|       1959|           34|     2936204.82|                    19|154537.095789473684|          2.8888888888888889|                     5|        0.6842105263157895|             0.2900094688|
|       2142|           30|     4408396.19|                    32|137762.380937500000|          1.9354838709677742|                     5|                      0.90625|             0.1418631999|
|       7880|           35|     1328334.23|                     7|189762.032857142857|          7.3333333333333333|                     5|        0.8571428571428571|             0.5898566131|
|      10817|           30|     2823423.23|                    18|156856.846111111111|          3.4705882352941178|                     5|        0.8888888888888888|             0.4046767831|
|       1580|           32|     2967152.59|                    19|156165.925789473684|          3.2777777777777777|                     5|        0.8947368421052632|             0.4479330536|
|       6658|           31|     6372935.18|                    37|172241.491351351351|          1.5277777777777777|                     5|        0.8378378378378378|             0.0814931810|
|       8592|           30|     1745499.70|                    14|124678.550000000000|          3.6153846153846154|                     5|        0.7857142857142857|             0.3105639033|
|       3175|           30|     3101957.47|                    17|182468.086470588235|                        3.5|                     5|        0.8235294117647058|             0.1750657852|
|       9852|           33|     5124076.50|                    36|142335.458333333333|          1.6571428571428573|                     5|        0.8333333333333334|             0.4754255562|
|       7982|           35|     2097029.81|                    13|161309.985384615385|          4.083333333333333|                     5|        0.9230769230769231|             0.9452366583|
|       9900|           47|     1723965.03|                     8|215495.628750000000|          5.285714285714286|                     5|                      0.875|             0.9219472219|
|      10623|           46|     1295904.95|                     9|143989.438888888889|                       4.75|                     5|                        1.0|             0.1365525149|
|        148|           31|     5580629.15|                    33|169109.974242424242|                       1.84375|                     5|        0.9090909090909091|             0.0940847270|
|       1645|           33|     4153795.99|                    23|180599.825652173913|          2.5454545454545454|                     5|        0.8695652173913043|             0.5340162601|
+-----------+-------------+---------------+----------------------+--------------------+---------------------------+----------------------+---------------------------+-------------------------+
only showing top 20 rows
```

Campo	Tipo	Descripción funcional	Cálculo / Origen	Ejemplo
`customer_id`	int	Identificador único del cliente.	`cus.customer_id` desde `dim_customer`, canal = `'tradicional'`.	`12345`
`recencia_dias`	int	Días de recencia: cuántos días han pasado desde el último pedido del cliente hasta el último día del mes actual.	`DATEDIFF(ped.month_end_date, ped.last_pedido_date)` donde `last_pedido_date = MAX(dat.date)` y `month_end_date = last_day(current_date())`	`12`
`ventas_total_3m`	decimal	Monto total neto vendido al cliente en los últimos 3 meses (suma de todas las líneas de pedido).	`SUM(lin.line_net_amount)` en la CTE `cte_customer_ped_3m` para los últimos 3 meses (`lin.period >= yyyymm hace 2 meses`).	`250000.50`
`frecuencia_pedidos_3m`	int	Número de pedidos distintos realizados por el cliente en los últimos 3 meses.	`COUNT(DISTINCT lin.order_id)` en `cte_customer_ped_3m`.	`8`
`ticket_promedio_3m`	decimal	Ticket promedio del cliente en los últimos 3 meses (monto promedio por pedido).	`ped.ventas_total_3m / ped.frecuencia_pedidos_3m`.	`31250.06`
`dias_promedio_entre_pedidos`	decimal	Días promedio entre pedidos consecutivos del cliente en los últimos 3 meses.	En `cte_customer_dias_promedio`: `LAG(order_date)` por `customer_id`; luego en `cte_avg_dias_promedio`: `AVG(datediff(order_date, last_date))`.	`9.3`
`variedad_categorias_3m`	int	Variedad de categorías de producto compradas por el cliente (número de categorías distintas) en los últimos 3 meses.	`COUNT(DISTINCT pro.product_category)` en `cte_customer_ped_3m`.	`4`
`porcentaje_pedidos_promo_3m`	decimal	Proporción de pedidos del cliente que tuvieron al menos una línea en promoción en los últimos 3 meses.	`ped.n_orders_promo / ped.frecuencia_pedidos_3m`, donde `n_orders_promo = COUNT(DISTINCT CASE WHEN lin.promo_flag = 1 THEN lin.order_id END)`. Valores entre `0` y `1`.	`0.25` (25%)
`mix_credito_vs_contado_3m`	decimal	Proporción de las ventas del cliente pagadas con crédito respecto al total vendido en los últimos 3 meses (mix por monto).	`ped.venta_credito_3m / ped.ventas_total_3m`, donde `venta_credito_3m = SUM(CASE WHEN lin.pay_type = 'credito' THEN lin.line_net_amount ELSE 0 END)`. Valores entre `0` y `1`.	`0.65` (65% crédito)

Nota: Todo el resultado final es bajo una ventana temporal de últimos 3 meses incluyendo el mes actual, definida por `lin.period >= CAST(date_format(add_months(current_date(), -2), 'yyyyMM') AS INT)`.

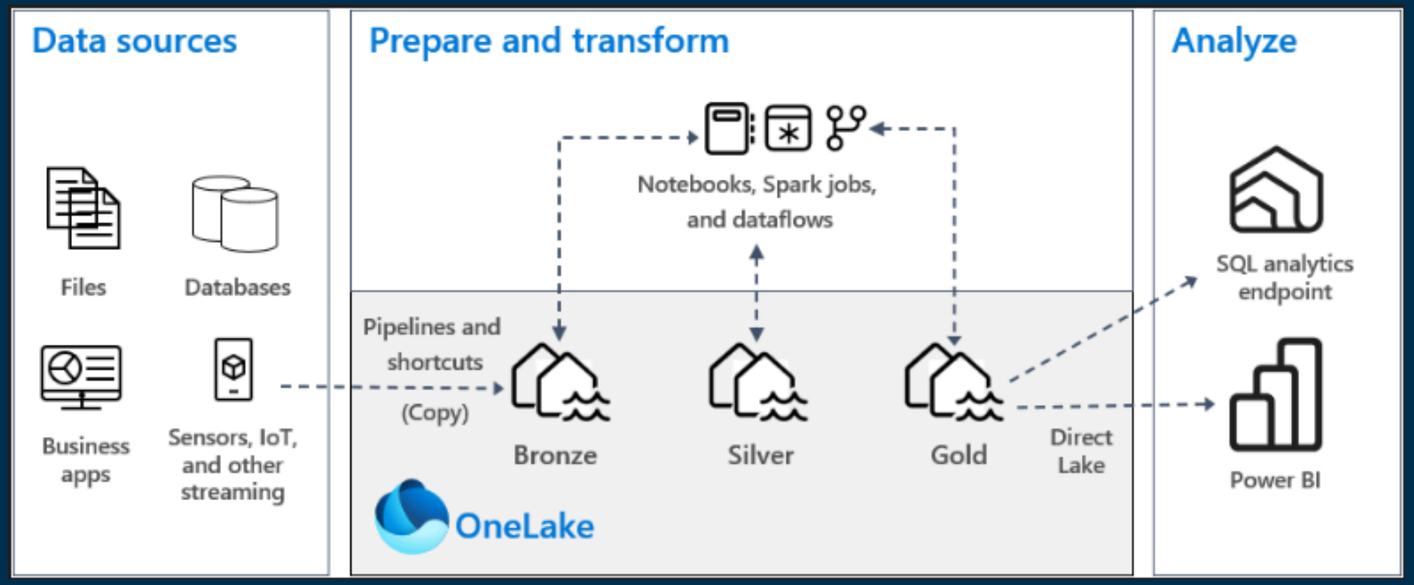
Solución Implementada

Ejecucion Pipeline




```python
import papermill as pm
from datetime import datetime

timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

bronze_pipelines = [
    ("../scripts/bronze/customer.ipynb", f"../scripts/bronze/outputs/bronze_customer_{timestamp}.ipynb"),
    ("../scripts/bronze/product.ipynb", f"../scripts/bronze/outputs/bronze_product_{timestamp}.ipynb"),
    ("../scripts/bronze/order_header.ipynb", f"../scripts/bronze/outputs/bronze_order_header_{timestamp}.ipynb"),
    ("../scripts/bronze/order_details.ipynb", f"../scripts/bronze/outputs/bronze_order_details_{timestamp}.ipynb")
]

silver_pipelines = [
    ("../scripts/silver/dim_customer.ipynb", f"../scripts/silver/outputs/silver_dim_customer_{timestamp}.ipynb"),
    ("../scripts/silver/dim_product.ipynb", f"../scripts/silver/outputs/silver_dim_product_{timestamp}.ipynb"),
    ("../scripts/silver/dim_date.ipynb", f"../scripts/silver/outputs/silver_dim_date_{timestamp}.ipynb"),
    ("../scripts/silver/fact_order_line.ipynb", f"../scripts/silver/outputs/silver_fact_order_line{timestamp}.ipynb"),
    ("../scripts/silver/fact_order.ipynb", f"../scripts/silver/outputs/silver_fact_order{timestamp}.ipynb")
]

gold_pipelines = [
    ("../scripts/gold/dim_features.ipynb", f"../scripts/gold/outputs/gold_dim_features{timestamp}.ipynb")
]

print("=== Iniciando ejecución automática de notebooks ===")

for nb_in, nb_out in bronze_pipelines:
    print(f"[RUN] {nb_in}")
    pm.execute_notebook(nb_in, nb_out)
    print(f"[OK]  Output: {nb_out}")

for nb_in, nb_out in silver_pipelines:
    print(f"[RUN] {nb_in}")
    pm.execute_notebook(nb_in, nb_out)
    print(f"[OK]  Output: {nb_out}")

for nb_in, nb_out in gold_pipelines:
    print(f"[RUN] {nb_in}")
    pm.execute_notebook(nb_in, nb_out)
    print(f"[OK]  Output: {nb_out}")

print("=== Pipeline completado ===")
```

Escalar a Microsoft Fabric



- Storage: ADLS Gen2
- Compatible con Delta Parquet (ACID)
- Spark Adminstrado (serverless)
- Integracion nativa con PowerBI
- Utilizacion de Notebooks para tratamiento de datos en Capas.
- Para ingestas Copy Data.
- Para pipelines Data Factory + Notebooks DAG ó Thirth Parties (ejemplo. AirFlow dentro de Fabric)
- Gobernanza incorporada en Fabric.

Sugerencias en Bronze:

- Capa previa (RAW, Landing) para recibir datos o archivos de origen.
- Utilizar Delta Parquet desde Bronzer en adelante.