Documentación Técnica – HACKATÓN DATA I

Genius Lab II – febrero 2026

Entregado a:

Bi Solutions - Aliware

Autor:

Marilyn Diana Ulloa Marcial

Rol:

Participante

Fecha de entrega:

06/02/2026

Contacto:

maryulloa6@gmail.com

LinkedIn

Índice

Resumen 3
Objetivo del proyecto y alcance 3
Fuentes de datos 3
Arquitectura 4
ETL 5
Modelo de datos 5
Dashboard 5
Ejecución 7
Conclusiones 13
Recomendaciones 13

Resumen

En este proyecto Hackatón Data se desarrolla un pipeline de integración de datos (ETL) en la herramienta Databricks Community Edition para transformar datos transaccionales de e-commerce en un dataset analítico y un dashboard gerencial orientado al área comercial, es decir ventas, clientes y productos.

La fuente principal es el dataset Online Retail de UCI, que es enriquecido con una segunda fuente vía API Frankfurter, los precios del dataset Online Retail están en GBP (sterling), para estandarizar el análisis, se convirtió cada monto a USD usando la API con tasas diarias GBP a USD, integradas por fecha de factura.

El resultado final incluye tablas por capas según la arquitectura Medallion (bronze/silver/gold), consultas por KPIs y un dashboard con visualizaciones construidas con herramientas nativas de Databricks.

Objetivo del proyecto y alcance

Implementar una ETL completa y entregar un dashboard comercial con insights accionables para soporte gerencial.

Fuentes de datos

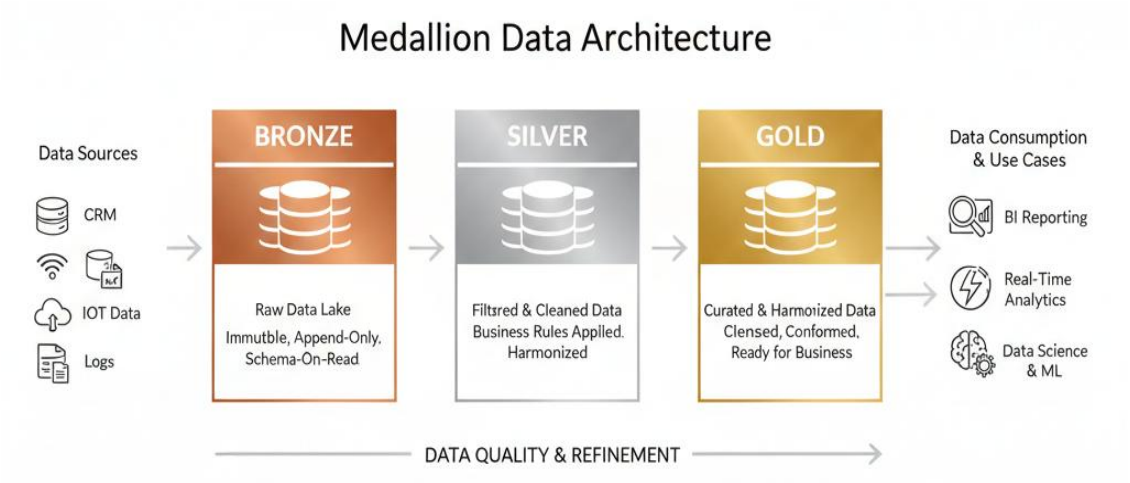
1. UCI – Online Retail
    - Tipo: Dataset transaccional de un retrailer online de Reino Unido.
    - Variables clave: InvoiceNo (si empieza con “c” indica cancelación), StockCode, Description, Quantity, InvoiceDate, UnitPrice (precio unitario en sterling/GBP), CustomerID, Country.
    - Licencia: Creative Commons Attribution 4.0, permite reutilización con atribución.

2. Frankfurter – Currency Data API
    - Tipo: API gratuita y open-source de tipos de cambio de referencia. No requiere API key ni tiene usage caps según su documentación.
    - Endpoints: latest, histórico por fecha y series de tiempo por rango
    - Consideración: Frankfurter alamecina fechas en UTC, por lo que la integración se hará por "fecha calendario".

Arquitectura

Se implementará una arquitectura tipo Medallion (Bronze/Silver/Gold) para mejorar progresivamente la calidad del dato y optimizar el consumo analítico.

- Bronze: Es la ingesta del Excel y captura de la respuesta de la API, preserva fidelidad y permite reprocesar.
- Silver: Es la limpieza, normalización, de duplicación, manejo de nulos, tipificación, y reglas de negocio (en este caso, tratamiento de cancelaciones por InvoiceNo con 'c').
- Gold: Son las agregaciones y modelos listos para BI (KPIs por día/cliente/producto), incluyendo conversión de montos desde GBP usando tabla de tipos de cambio por fecha.



Realizado por: Marilyn Ulloa

La capa Gold alimentará el dashboard gerencial, construido con visualizaciones nativas disponibles en notebooks o SQL editor de Databricks

ETL

Se implementó un pipeline ETL en Databricks Community Edition siguiendo el enfoque Medallion para asegurar trazabilidad y mejorar la calidad de los datos progresivamente.

En Bronze, se almacenan los datos crudos: bronze_online_retail (ingesta del dataset transaccional) y bronze_fx_gbp_usd (captura de tasas GBP-USD provenientes de la API Frankfurter) preservando fidelidad para reprocesos.

En Silver, se realiza limpieza y estandarización: silver_online_retail aplica tipificación, manejo de nulos/duplicados y reglas de negocio (cancelaciones identificadas por InvoiceNo con prefijo "C"); silver_fx_gbp_usd normaliza la tabla de tipos de cambio; y silver_online_retail_usd integra la tasa por fecha de factura para convertir importes de GBP a USD.

En Gold, se generan agregaciones listas para consumo analítico:

gold_kpi_daily, gold_kpi_country, gold_kpi_daily_country y gold_kpi_product_top, diseñadas para responder preguntas de negocio (ventas, devoluciones, órdenes, clientes y desempeño por país/producto) y alimentar el dashboard.

Modelo de datos

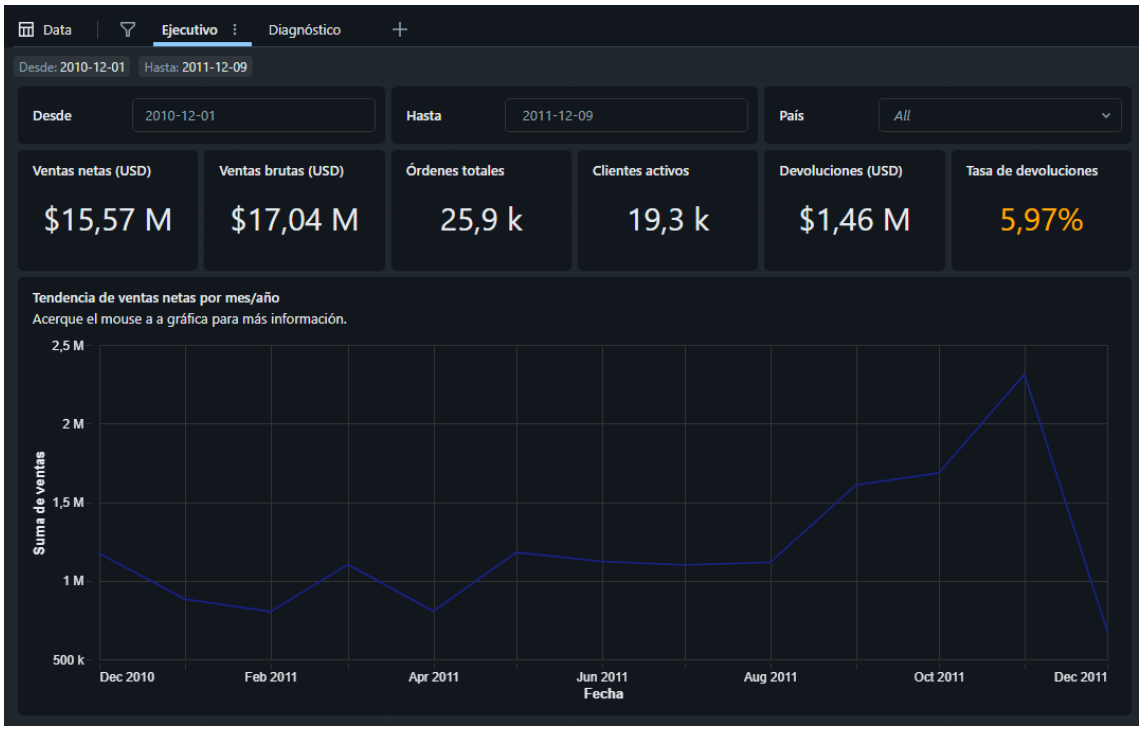
El modelo analítico se basa en tablas agregadas en capa Gold para optimizar consultas y visualización. Las entidades principales son: KPIs por día (gold_kpi_daily), KPIs por país (gold_kpi_country), KPIs por día y país (gold_kpi_daily_country) y ranking de productos (gold_kpi_product_top), todas derivadas de la transaccional enriquecida en USD (silver_online_retail_usd).

La tabla silver_fx_gbp_usd actúa como dimensión/lookup de tipo de cambio por fecha, garantizando consistencia en la conversión GBP-USD usada en las agregaciones.

Dashboard

Se construyó un dashboard gerencial con visualizaciones nativas de Databricks orientado al área comercial, con dos vistas:

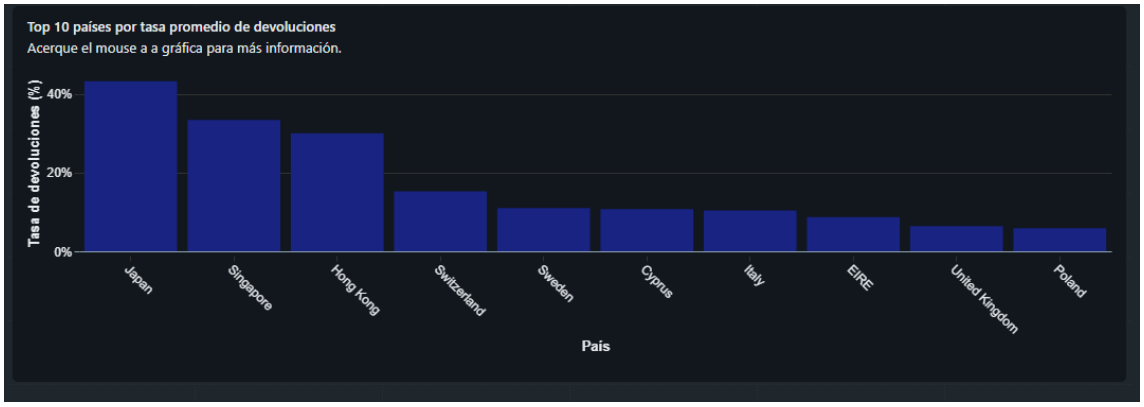
- La vista Ejecutivo presenta KPIs principales (ventas netas, ventas brutas, devoluciones, órdenes, clientes activos y tasa de devoluciones) y una tendencia temporal para seguimiento del desempeño.



Realizado por: Marilyn Ulloa

- La vista Diagnóstico profundiza en el análisis por país (Top por ventas netas, tendencia de devoluciones y Top por tasa de devoluciones) para identificar mercados con mejor desempeño y focos de alerta por devoluciones.

Realizado por: Marilyn Ulloa



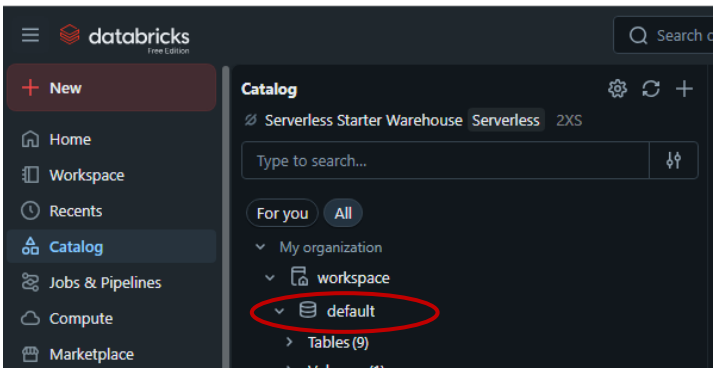
Realizado por: Marilyn Ulloa

Ejecución

Para reproducir el proyecto completo desde cero en Databricks Community Edition, sigue estos pasos:

1. Configuración inicial

- Crear un workspace en Databricks Community Edition y activar un cluster Serverless o starter.
- Crear un schema/database llamado default o el nombre que prefieras, donde se almacenarán las 9 tablas del pipeline.

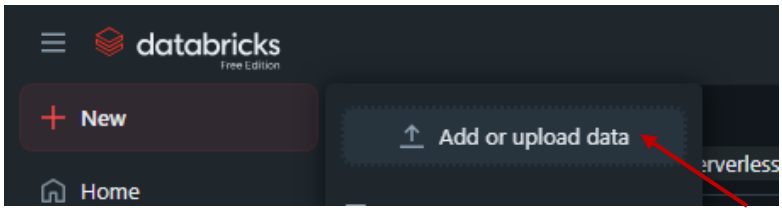


Realizado por: Marilyn Ulloa

2. Ingesta de datos (capa Bronze)

- **Tabla bronze_online_retail**:

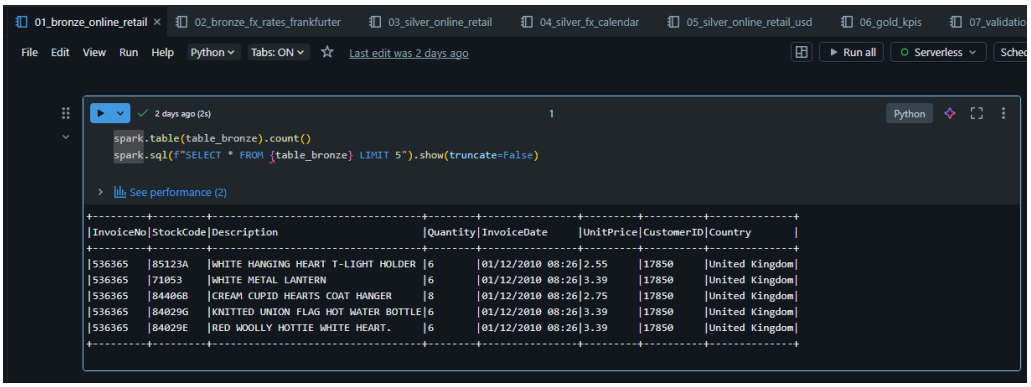
  Subir el archivo Excel del dataset UCI Online Retail a Databricks (opción "Upload Data" o desde notebook con spark.read.format("excel")), conservando todas las columnas sin transformación (InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, Country).



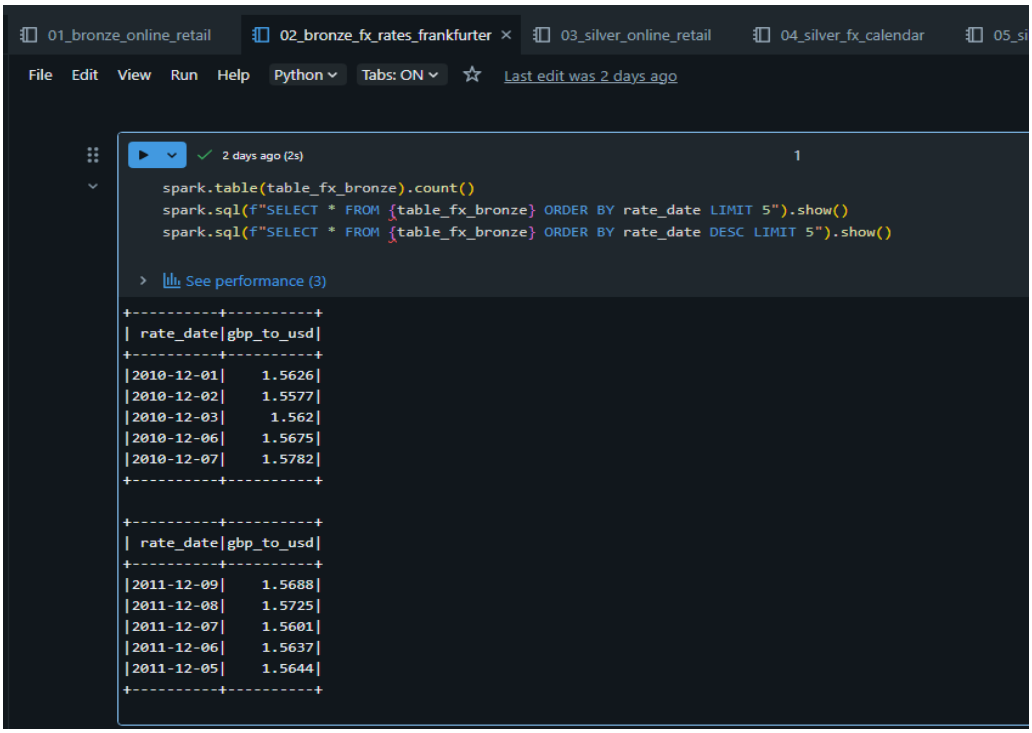
Realizado por: Marilyn Ulloa

- **Tabla bronze_fx_gbp_usd**:

  Ejecutar un script Python o notebook que llame a la API Frankfurter con el endpoint de series históricas (por ejemplo, /timeseries?start_date=YYYY-MM-DD&end_date=YYYY-MM-DD&from=GBP&to=USD) para obtener las tasas diarias GBP→USD del rango de fechas del dataset transaccional, y almacenar en formato tabla con columnas date, rate.



Realizado por: Marilyn Ulloa

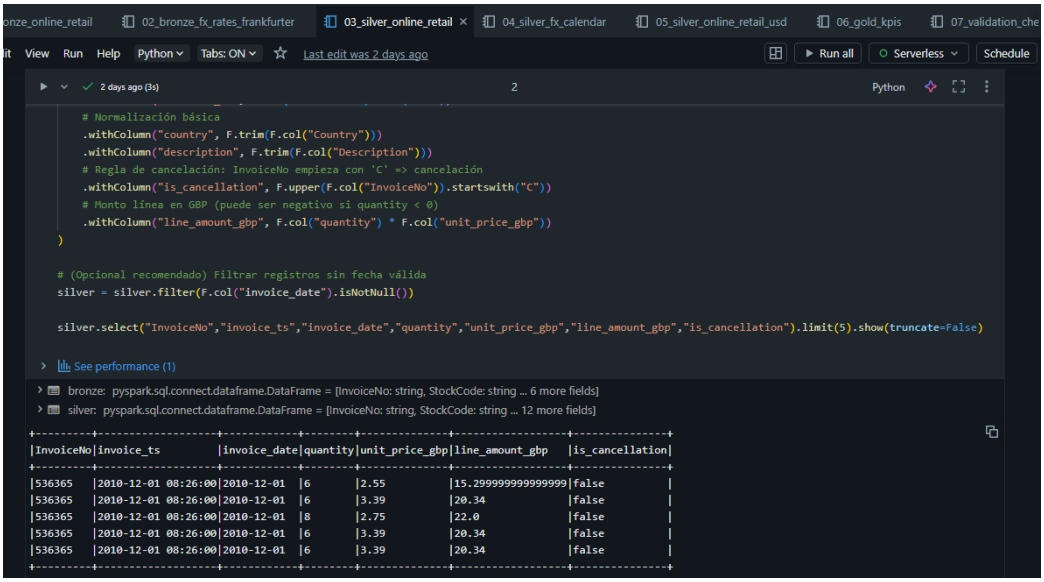


Realizado por: Marilyn Ulloa

3. Transformación y limpieza (capa Silver)

- **Tabla silver_online_retail**:

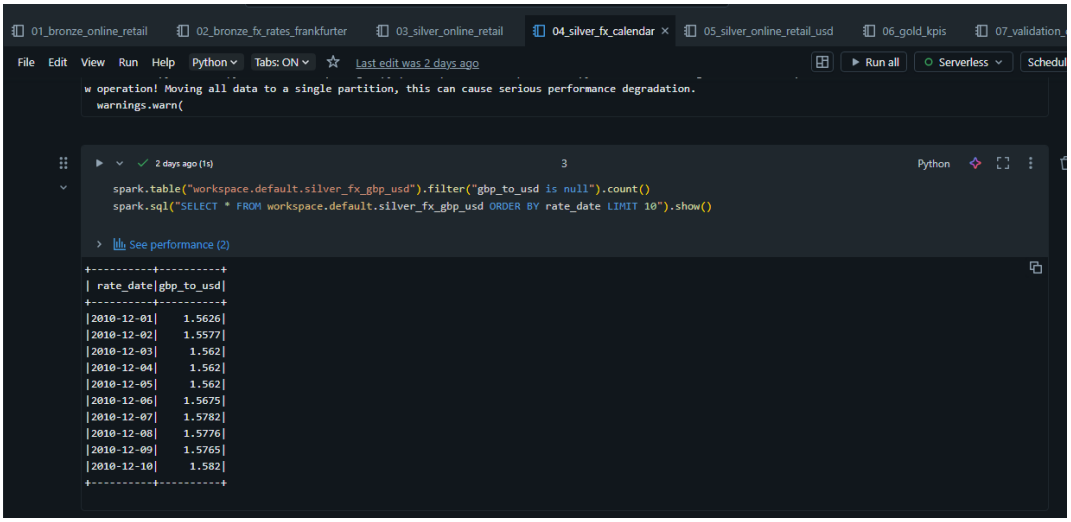
  Aplicar limpieza con SQL o PySpark: eliminar duplicados, filtrar nulos en campos críticos (InvoiceNo, CustomerID, Quantity > 0 para ventas válidas), identificar y marcar cancelaciones (InvoiceNo que inicia con "C"), y tipificar columnas (fecha como timestamp, montos como decimal).



Realizado por: Marilyn Ulloa

- **Tabla silver_fx_calendar**:

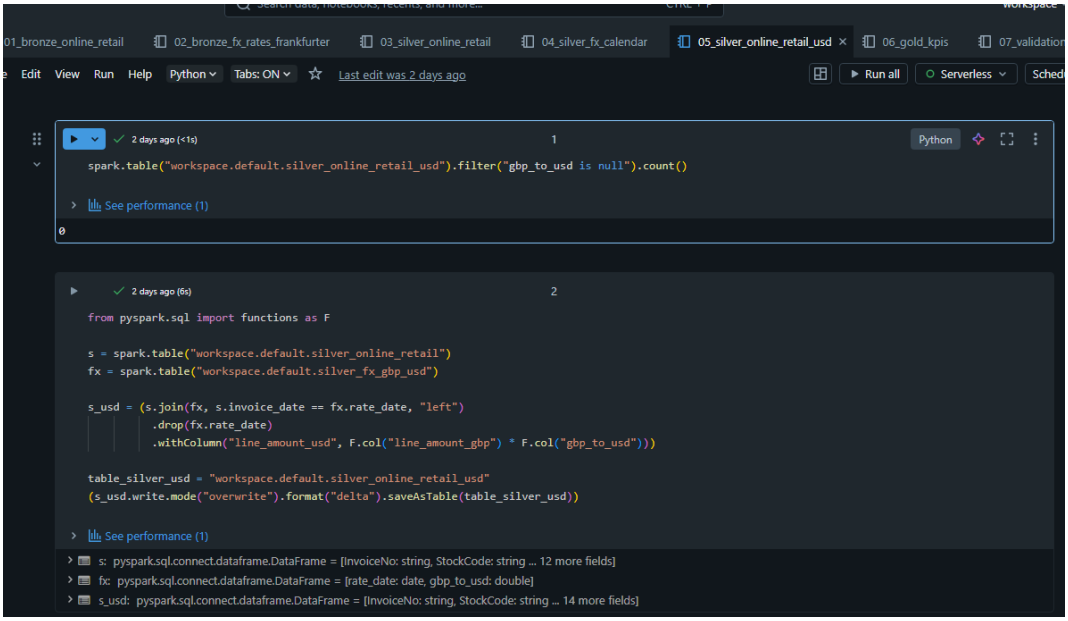
  Normalizar la tabla de tipos de cambio asegurando un único rate por fecha (manejar duplicados si existen), y formatear date como tipo Date para facilitar joins.



Realizado por: Marilyn Ulloa

- **Tabla silver_online_retail_usd**:

  Hacer un join de silver_online_retail con silver_fx_gbp_usd por fecha de factura, y crear columnas calculadas UnitPrice_USD = UnitPrice_GBP * rate y TotalAmount_USD = Quantity * UnitPrice_USD, obteniendo así todas las transacciones en moneda estandarizada.



Realizado por: Marilyn Ulloa

4. Agregaciones analíticas (capa Gold)

Ejecutar queries de agregación para generar las 4 tablas Gold optimizadas para BI:

- **gold_kpi_daily**: Agrupar por fecha (día) y calcular

  SUM(gross_sales_usd), SUM(returns_usd), net_sales_usd = gross - returns,

  COUNT(DISTINCT orders), COUNT(DISTINCT customers),

  AVG(return_rate).

- **gold_kpi_country**: Agrupar por país y calcular las mismas métricas agregadas por país.

```
✓ 2 days ago (4s)                                                    3

gold_country = (s.groupBy("country")
  .agg(
    F.sum("line_amount_usd").alias("sales_usd_net"),
    F.countDistinct("InvoiceNo").alias("orders"),
    F.countDistinct("customer_id").alias("customers")
  )
  .withColumn("aov_usd", F.col("sales_usd_net") / F.col("orders"))
  .orderBy(F.desc("sales_usd_net"))
)

gold_country.write.mode("overwrite").format("delta").saveAsTable("workspace.default.gold_kpi_country")

> See performance (1)
> gold_country: pyspark.sql.connect.dataframe.DataFrame = [country: string, sales_usd_net: double ... 3 more fields]
```

Realizado por: Marilyn Ulloa

- **gold_kpi_daily_country**: Agrupar por día y país (granularidad más fina para análisis por mercado en el tiempo).

```
✓ 2 days ago (5s)                                                    4

from pyspark.sql import functions as F

s = spark.table("workspace.default.silver_online_retail_usd")

gold_daily = (s.groupBy("invoice_date")
  .agg(
    F.sum("line_amount_usd").alias("sales_usd_net"),
    F.countDistinct("InvoiceNo").alias("orders"),
    F.countDistinct("customer_id").alias("active_customers"),
    F.sum(F.when(F.col("is_cancellation"), 1).otherwise(0)).alias("cancelled_lines")
  )
  .withColumn("aov_usd", F.col("sales_usd_net") / F.col("orders"))
  .orderBy("invoice_date")
)

gold_daily.write.mode("overwrite").format("delta").saveAsTable("workspace.default.gold_kpi_daily")

> See performance (1)
```

Realizado por: Marilyn Ulloa

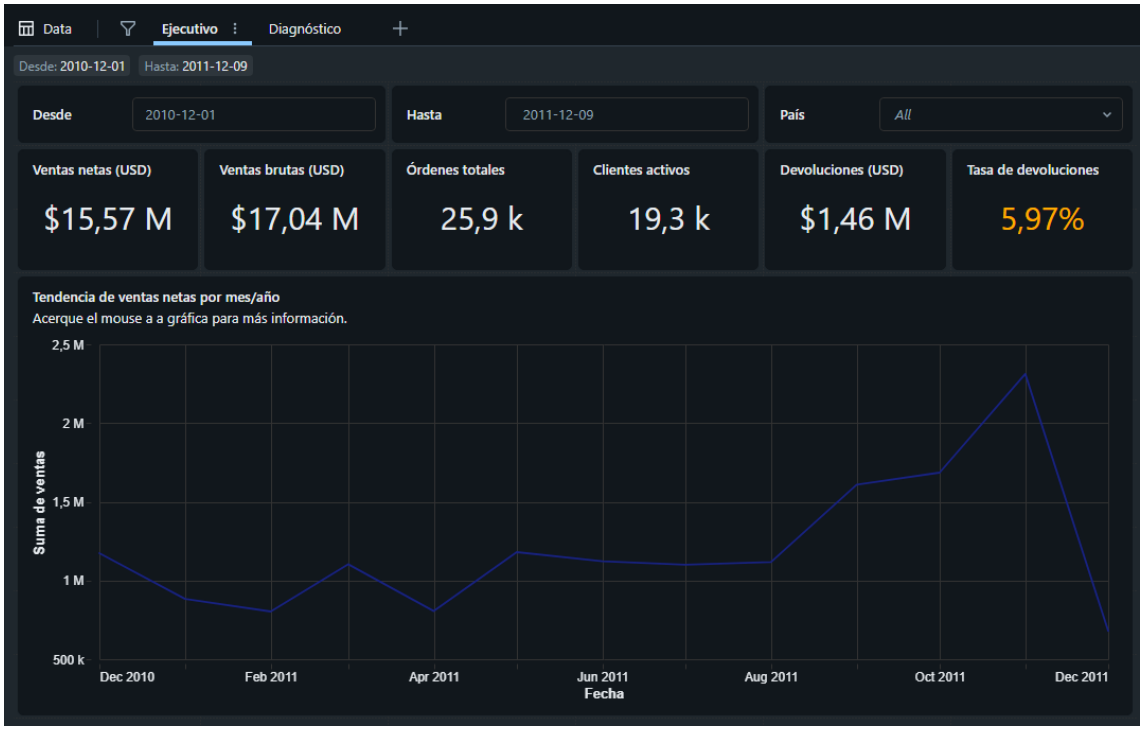
- **gold_kpi_product_top**: Agrupar por producto (StockCode/Description) y calcular ventas, ordenar descendente y limitar a Top N (por ejemplo, Top 10).

```
✓ 2 days ago (4s)                                                    2

gold_product = (s.groupBy("StockCode", "description")
  .agg(
    F.sum("line_amount_usd").alias("sales_usd_net"),
    F.sum("quantity").alias("units_net"),
    F.countDistinct("InvoiceNo").alias("orders")
  )
  .orderBy(F.desc("sales_usd_net"))
)

gold_product.write.mode("overwrite").format("delta").saveAsTable("workspace.default.gold_kpi_product_top")

> See performance (1)
> gold_product: pyspark.sql.connect.dataframe.DataFrame = [StockCode: string, description: string ... 3 more fields]
```

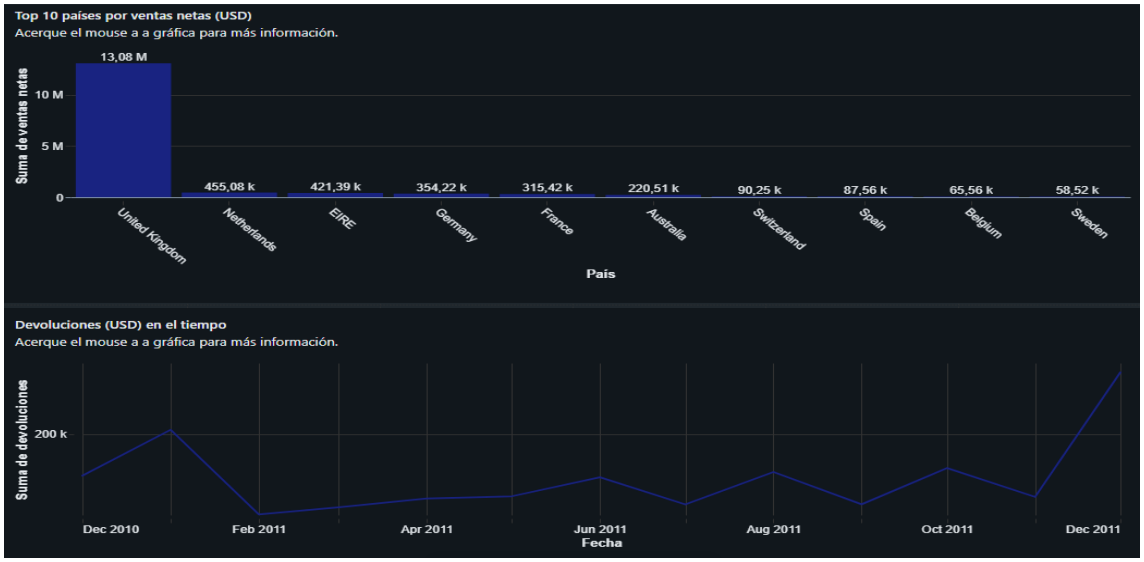
Realizado por: Marilyn Ulloa

5. Construcción del dashboard

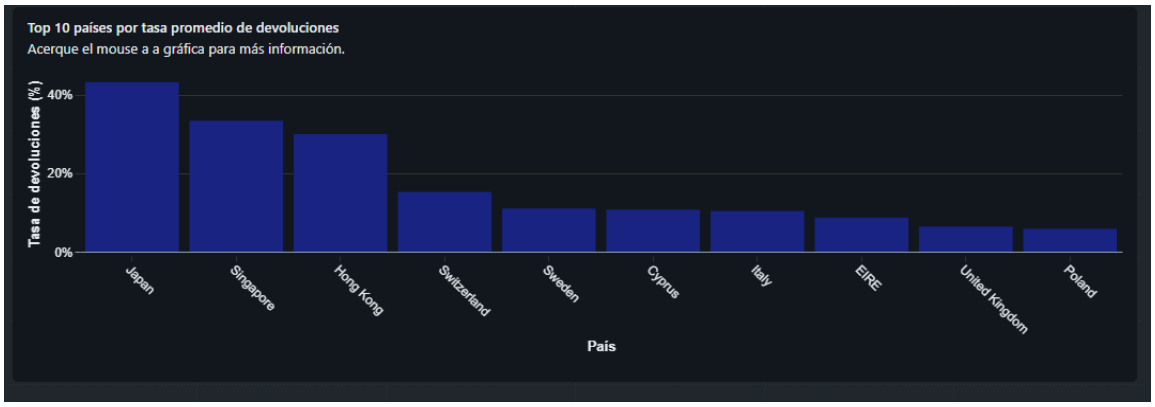
- Desde SQL Editor o Dashboard de Databricks, crear consultas sobre las tablas Gold (gold_kpi_daily, gold_kpi_country, gold_kpi_daily_country) para alimentar los KPIs y gráficos.
- Configurar visualizaciones: tarjetas (Counters) para KPIs principales, line chart para tendencia temporal, bar charts para Top 10 países por ventas y tasa de devoluciones.
- Aplicar filtros de fecha (desde/hasta) y país (dropdown "All" o selección individual) que afecten todas las visualizaciones para interactividad.



Realizado por: Marilyn Ulloa



Realizado por: Marilyn Ulloa



Realizado por: Marilyn Ulloa

Conclusiones

El proyecto demuestra la implementación exitosa de un pipeline ETL completo en Databricks Community Edition, logrando transformar datos transaccionales crudos en un dashboard gerencial accionable con arquitectura Medallion.

- Integración de fuentes externas: La combinación del dataset UCI Online Retail con la API Frankfurter para conversión GBP-USD permite análisis estandarizado en una moneda única, eliminando distorsiones por tipo de cambio.
- Escalabilidad y buenas prácticas: El enfoque Medallion garantiza trazabilidad, permite reprocesos sin perder datos crudos y optimiza costos al separar transformación pesada de consultas analíticas ligeras.
- Valor para negocio: El dashboard entrega insights inmediatos (ventas netas, tasa de devoluciones, ranking de países) que soportan decisiones gerenciales sobre mercados prioritarios, gestión de devoluciones y planificación comercial.
- Eficiencia de recursos: El uso de Databricks Community Edition, API Frankfurter y capas de agregación Gold minimiza costos de infraestructura y tiempos de respuesta del dashboard.

Recomendaciones

Para siguientes fases del proyecto o proyectos similares, se sugieren las siguientes mejoras:

- Automatización del pipeline: Implementar jobs o workflows en Databricks para ejecutar el ETL de forma programada de manera diaria o semanal, incluyendo

refresh automático de las tasas de cambio desde Frankfurter y actualización incremental de las capas silver/gold.

- Enriquecimiento con más dimensiones: Integrar datos de inventario, costos logísticos o información demográfica de clientes para análisis de rentabilidad por producto/país y segmentación.
- Gobernanza y seguridad: Aplicar políticas de acceso granular en las tablas gold si se maneja información sensible, y documentar linaje de datos desde bronze hasta dashboard para auditorías y cumplimiento normativo.