Quales

Data Management y Data Visualización

Ejercicio Integrador Final

Preparado por Melina Lugones 30/1/2026

Índice

Archivos Entregados:

Carpeta **Scripts** Python y SQL necesarios para crear y cargar el DataWarehause

• orquestador.py – archivo de Python que al ser ejecutado corre todos los scripts de

• SQLQuerySTAGING.sql – Script para crear las tablas STAGING

• SQLQueryINT.sql – Script para crear las tablas INT

• SQLQueryCreateDW.sql – Script para crear las tablas finales del DW

• SQLQueryStoreProcedures.sql – Script para crear los StoreProcedures necesarios

• extract_data.py – Script de Python para Extraer los datos de los archivos CSV y

cargarlos en las tablas Staging

• load_STG_to_INT.py – Script de Python para Cargar las tablas Int a partir las Staging

• dw_loader.py - Script de Python para Cargar las tablas finales del DW a partir de las Int

• config.ini – archivo con la configuración del servidor y la base de datos para que los

scripts puedan acceder a ella

• Modelado.xlsx – archivo con el detalle del modelado del DW

• DataShop_Dashboard.pbix – archivo de Power BI con el tablero para visualizar los datos

• Carpeta “DATASET” – carpeta que contiene los archivos CSV de donde los Scripts de

Python toman los datos

• Carpeta “generar registros” – carpeta que contiene el Script de Python que se utilizo

para generar los archivos CSV, al ejecutarlo los generan en esta carpeta

• Carpeta “SVG Backgrounds “ – carpeta con los archivos SVG para la construcción del

Tablero en Power BI

Análisis inicial

Datos en los archivos CSV proporcionados.

Estructura del Modelo:

Los datos están organizados en un modelo de **Estrella/Copo de Nieve** :

- **Hechos:** ventas y entregas.
- **Dimensiones:** clientes, productos, tiendas, almacenes y estado del pedido.

Vínculos detectados:

- ventas se conecta con productos, clientes y tiendas.
- entregas se conecta con ventas (vía CodVenta), almacenes y estado del pedido.

Hallazgos : Datos Nulos e Inconsistencias

Se identificaron fallas críticas en la integridad de los datos que deben resolverse durante el proceso del ETL antes del Análisis, por ejemplo:

- **Productos:** El producto ID 12 es un duplicado del ID 1, pero le falta la Marca. El ID 13 no tiene PrecioCosto y el ID 14 no tiene PrecioVentaSugerido ni Categoría.
- **Tiendas:** La tienda ID 14 no tiene Nombre (Descripción). Existen nulos en Dirección, Localidad, CP y TipoTienda.
- **Entregas vs. Estados:** Hay una inconsistencia lógica grave. En el archivo entregas, se asignan estados que no coinciden con la tabla maestra. Por ejemplo:

    - CodEstado 1 en la maestra es "En preparación", pero en entregas aparece asociado a "Devuelto" (ID 1) o "Entregado" (ID 7). Los IDs de estado en la tabla de entregas están rotos o mal mapeados.

- **Desfase de Fechas:** La entrega ID 16 tiene Fecha_Envio "2024-01-01", pero su venta asociada (ID 16) ocurrió el "2024-02-16". Es imposible enviar un producto antes de que se venda.
- **Precios Inconsistentes:** En ventas, el producto "Smartwatch" (ID 10) se vende a 200.00, cuando el PrecioVentaSugerido en la tabla de productos es 150.00. Esto podría ser una política de precios dinámica o un error de carga.
- **PrecioVenta no coincide con precios sugeridos:** por ejemplo el Microondas: costo 80, sugerido 100 → vendido a 200**.** La aspiradora: sugerido 250 → vendida a 150**.**
- **Redundancia peligrosa:**
    - CodigoProducto y Producto
    - CodigoCliente y Cliente
    - CodigoTienda y Tienda

Relaciones entre tablas (y problemas)

RELACION ESPERADA	ESTADO
Ventas → Clientes	Conceptual, pero mal implementada
Ventas → Productos	Rota por duplicación
Ventas → Tiendas	Rota por redundancia
Ventas → Entregas	NO trazable
Entregas → Estados	Inconsistente
Entregas → Almacenes	Parcial

Reporte de negocio a primera vista: ¿Qué nos dicen los datos?

Dada la escasa cantidad de registros se aprovecha para hacer un mínimo análisis de los datos que sirven para mas adelante comparar con los gráficos finales del ETL y asegurar su veracidad.

- **Logística Bajo Presión:** De las 20 entregas registradas, 5 terminaron en "Devuelto" (25% de tasa de devolución). Es un KPI alarmante que sugiere problemas en la calidad del producto o en la última milla.
- **Eficiencia de Proveedores:** *FastDelivery* tiene tiempos de entrega muy erráticos (desde 4 días hasta más de un mes). *ShipPro* parece ser el más consistente en entregas rápidas.
- **Concentración de Clientes:** Empresas como **Globex**, **Initech** y **ACME** son compradores recurrentes de artículos de alto valor (Televisores, Lavadoras), representando el mayor volumen de facturación.
- **Omnicanalidad:** El canal "Online" (Tiendas 1, 7, 9) tiene un flujo constante, pero las sucursales físicas como "ElectroShop California" están moviendo mucho volumen de gadgets pequeños (Smartwatches).

Desafíos Identificados para el Análisis:

Para transformar estos CSV en un Data Warehouse funcional, estos datos presentan los siguientes retos:

- **Calidad de datos:** Nulos críticos**,** Duplicados**,** Inconsistencias semánticas.
- **Fechas claves:** pedidos figuran entregados antes de la fecha de venta. Esto afecta los KPIs de eficiencia logística.
- **Falta de claves confiables:** No hay IDs únicos consistentes entre hechos**,** Relaciones no garantizadas.
- **Redundancia:** Texto + código en múltiples tablas**,** Alto riesgo de divergencia histórica.
- **Gobierno de datos inexistente:** No hay catálogos maestros**,** No hay reglas de validación**,** No hay control de integridad-
- **Análisis financiero limitado:** Costos incompletos**,** Precios no confiables**,** Margen imposible de calcular correctamente.

Se identifico que en el ETL, entre otras cosas, se debe decidir entre confiar en el CodEstado numérico o en la descripción de texto, ya que actualmente no coinciden. También se debe corregir las fechas de envío que anteceden a las de venta para no arruinar las métricas de *Lead Time*.

También que es necesario completar los valores nulos en marcas y categorías de productos para poder agrupar las ventas por rubro de forma precisa y validar si las diferencias entre PrecioVenta y PrecioSugerido son descuentos/recargos intencionales o ruido en los datos.

Planteo del Modelado:

El Data Warehouse fue diseñado bajo los principios de modelado dimensional de Kimball, utilizando un esquema en estrella extendido. Esta arquitectura destaca por la implementación de dos tablas de hechos centrales, **Fact_Ventas** y **Fact_Entregas**, lo que permite separar el evento comercial transaccional del proceso logístico posterior.

En cuanto a la estructura de las relaciones, el modelo aplica una cardinalidad de muchos-a-uno para el proceso de ventas, donde cada transacción se vincula a un único producto, cliente y tienda a través de claves subrogadas (ID_Producto, ID_Cliente, etc.). Para el proceso logístico, la relación es de uno-a-muchos, contemplando que una sola venta puede derivar en múltiples eventos de entrega, como reenvíos o entregas parciales.

Un aspecto fundamental del desarrollo fue el enriquecimiento de los datos originales. Se incorporaron dimensiones con atributos avanzados que no existían en los archivos planos, como precios de costo en **Dim_Producto** y métricas de desempeño en **Dim_Proveedor** (costo por kilómetro y tiempos promedio). Asimismo, se añadieron columnas técnicas de auditoría como FechaCarga y campos calculados como el Total_IVA y el CostoEntrega, y una Tabla para almacenar registros rechazados.

Se incluyo una columna para almacenar el tiempo de envio y otra para el tiempo de entrega (Tiempo_Key_Envio y Tiempo_Key_Entrega) para calcular con precisión el *Lead Time* y detectar retrasos.

Evaluación de las reglas de negocio (RN)

RN-01: Tiempo de Entrega : diferencia entre fecha de envío y fecha de entrega real.
El modelo contiene: Tiempo_Key_Envio y Tiempo_Key_Entrega. Ambos referencian Dim_Tiempo, lo que permite calcular el tiempo de entrega mediante una diferencia entre fechas sin ambigüedad. Por lo tanto la regla no está almacenada como métrica, pero el modelo provee toda la información necesaria para calcularla en vistas.

RN-02: Costo Total de la Entrega: acumulado en función del proveedor y la distancia del envío.

Para cumplir con esto se decidió agregar el dato de DistanciaKM, aun que los datos para calcular esto no se encuentran en los CVS originales, el ETL ya tiene planteado en el diseño lo necesario para calcularlo cuando estos datos se incorporen.

Fact_Entregas incluye CostoEntrega, DistanciaKM y ID_Proveedor lo que permite analizar el costo total por entrega, analizar por proveedor y validar o recalcular los costos en función de la distancia

RN-03: Entregado a tiempo**:** fecha de entrega real menor o igual a la fecha estimada. Dado que el modelo incluye los datos de Tiempo_Key_Entrega y FechaEstimada se pude calcular fasilmente.

RN-04: Costo por kilómetro**:** costo total de la entrega dividido por la distancia recorrida. Dado que Fact_Entregas cuenta con CostoEntrega y DistanciaKM esto se puede calcular fasilmente.

Desarrollo del ETL

El ETL esta estructurado en tres capas bien diferenciadas: STG (staging), INT (integración) y DW (data warehouse). Cada una tiene una responsabilidad específica y los Store Procedures actúan como puntos de control de calidad entre capas.

La capa STG funciona como zona de aterrizaje de datos crudos. Allí se asume que los datos pueden venir incompletos, mal formateados, duplicados o con valores inválidos. El objetivo es aislar el impacto del origen y no contaminar las capas posteriores.

En la capa INT los Store Procedures STG_to_INT aplican sistemáticamente limpieza y normalización de datos. Se usa LTRIM y RTRIM para eliminar espacios, UPPER y LOWER para

estandarizar claves y descripciones, NULLIF para convertir strings vacíos en valores nulos reales y TRY_CAST para evitar que errores de formato interrumpan la ejecución.

Cada SP valida los campos obligatorios en la cláusula WHERE antes de insertar en INT. Los registros que no cumplen esas condiciones se registran en la tabla ETL_Registros_Rechazados junto con el motivo del rechazo y el ID del proceso. Esto permite auditar la calidad del dato, analizar patrones de error y mejorar el origen sin que el proceso completo falle.

La utilización de SELECT DISTINCT en las cargas hacia INT evita la duplicación de registros cuando el origen trae datos repetidos. Evitando que duplicados compliquen el análisis más adelante.

Cuando el origen no provee ciertas columnas necesarias para el modelo analítico, el ETL adopta una estrategia de valores por defecto. Esto evita nulos en columnas métricas, mantiene la integridad del esquema y deja abierta la posibilidad de enriquecer esos datos en el futuro sin romper el pipeline.

En Entregas se valida la coherencia temporal asegurando que la fecha de entrega no sea anterior a la fecha de envío.

Todos los Store Procedures de STG a INT están encapsulados en transacciones explícitas. Esto garantiza atomicidad: o se carga la totalidad de los datos válidos o no se carga nada. Si ocurre un error, la tabla de integración no queda en un estado intermedio.

El SP_Orquestador_STG_to_INT coordina la ejecución de todos los procesos en el orden correcto, respetando dependencias entre entidades. Las dimensiones base se cargan antes que los hechos, y las tablas de las que dependen otras, como Estados, Almacenes o Proveedores, se procesan antes de Ventas y Entregas.

Además, el orquestador registra cada ejecución en la tabla ETL_Control_Procesos, almacenando fecha de inicio, fecha de fin, estado, cantidad de registros procesados y rechazados.

En la capa INT a DW, los Store Procedures trabajan con una lógica de UPSERT para las dimensiones. Si un registro ya existe, se actualizan sus atributos descriptivos; si no existe, se inserta uno nuevo. Esta estrategia permite que las dimensiones evolucionen en el tiempo sin duplicarse y sin perder consistencia, incluso cuando cambian descripciones, categorías o atributos secundarios.

En la capa INT se trabaja con claves naturales provenientes del origen. En la capa DW, las dimensiones generan claves surrogate internas. Esta decisión protege a los hechos de cambios en el origen y es fundamental para la estabilidad del modelo dimensional.

Las dimensiones como Dim_Almacen y Dim_Tienda separan geográficamente los datos (Localidad, Provincia, Ciudad), permitiendo reportes de ventas por zona incluso cuando la dirección exacta en el CSV era nula o parcial.

Se aplica una normalización del texto con UPPER(LTRIM(RTRIM(Campo))). Esto elimina espacios accidentales y unifica mayúsculas para que "Samsung", "samsung" y " SAMSUNG " se traten como la misma entidad.

El SP_STG_to_INT_EstadoPedido soluciona la inconsistencia de los estados mediante reglas de patrón: Se usa CASE WHEN LIKE '%entregad%'... para crear una columna nueva llamada Tipo_Estado (Completado, Cancelado, En Proceso). Esto permite que, aunque el CSV sea inconsistente, en el analisis se pueda agrupar datos fácilmente. También se asigna un Orden_Secuencia lógico (1 para Preparación, 5 para Entregado). Esto permite crear embudos de conversión y entender cuánto tiempo pasa un pedido en cada etapa.

Se usa TRY_CAST y REPLACE(Precio, ',', '.'). Esto evita que el ETL se rompa si un precio viene con coma en lugar de punto o con caracteres no numéricos.

Y para la validación de Fechas en SP_STG_to_INT_Entregas, se incluyó una regla de oro: AND (Fecha_Entrega >= Fecha_Envio) que soluciona el problema detectado donde un envío parecía viajar al pasado. Si la fecha es incoherente, el registro se carga en **la tabla de registros descartados** y no se carga en la tabla de Dim_Entregas.

En la etapa final hacia el DW, los SPs usan una lógica de Update + Insert, entonces si el código ya existe, pero cambió la descripción o el estado, actualiza los datos existentes y si el código es nuevo, lo agrega. Así garantiza que el Data Warehouse siempre está actualizado sin duplicar registros.

Durante la creación del ETL surgió un problema en la carga de los datos, a pesar de que el orquestador y todos los scripts corrían perfectamente, la tabla de entregas en el DW quedaba bacía. Después de correr Queries de diagnósticos se encontró que los datos si estaban correctamente cargados en las tablas INT, y que la falta de datos en FACT_Entregas se debía a un desajuste de claves entre el origen de datos y el destino. Existía una colisión lógica entre la Clave Natural (Surrogate Key candidata en el CSV) y la Clave Surrogada (PK Autoincremental en el DW):

- **Origen (CSV/INT):** CodVenta es un identificador secuencial relativo al archivo ($1, 2, 3...$).
- **Destino (Fact_Ventas):** ID_Venta es una PK IDENTITY que, por cargas previas o truncados, inició en un valor distinto ($42, 43, 44...$).
- **Conflicto de Integridad:** Al intentar insertar en Fact_Entregas usando el CodVenta del CSV, la **Constraint de Foreign Key** fallaba o devolvía un conjunto vacío porque el valor 1 no existía en el dominio de la columna ID_Venta de la tabla de hechos.

Se confirmó que había un error en el truncate, para proteger las claves no se estaba ejecutando. Para resolverlo se agregó una función de limpieza para las tablas en Python, en el orquestador principal del ETL completo, así se logró que las claves autoincreméntales de SQL no se acumulen cada vez que se corre el programa.

Además de esto se implementó un mapeo por posición lógica que asegura que si en el futuro SQL Server vuelve a saltar un número (algo que a veces pasa por errores de red o reinicios del servidor), el ETL no se va a romper. El ROW_NUMBER() siempre unirá la "Venta 1 del CSV" con la "Venta 1 de SQL", sin importar si el ID se llama 1, 42 o 100.

Al finalizar el proceso de creación del ETL se corrieron algunas **Queries de auditoria** antes de pasar a crear el dashboard, por ejemplo las siguientes:

Mapa de Entregas por Almacén y Cliente: Esta query permite ver de qué almacén está saliendo la mercadería para cada cliente.

```sql
SELECT
    da.NombreAlmacen,
    dc.RazonSocial AS Cliente,
    COUNT(fe.ID_Entrega) AS Cantidad_Envios,
    SUM(fe.CantidadProductos) AS Total_Items_Enviados
FROM Fact_Entregas fe
JOIN Dim_Almacen da ON fe.ID_Almacen = da.ID_Almacen
JOIN Dim_Cliente dc ON fe.ID_Cliente = dc.ID_Cliente
GROUP BY da.NombreAlmacen, dc.RazonSocial
ORDER BY da.NombreAlmacen, Cantidad_Envios DESC;
```

Análisis de Tiempo de Entrega (KPI Logístico): Para medir cuántos días pasan desde el envío hasta la entrega real, agrupado por proveedor. Esto verifica que la relación con Dim_Tiempo sea sólida.

```sql
SELECT
    dp.NombreProveedor,
    AVG(DATEDIFF(DAY, t_envio.Fecha, t_entrega.Fecha)) AS
Promedio_Dias_Entrega,
    MIN(DATEDIFF(DAY, t_envio.Fecha, t_entrega.Fecha)) AS Entrega_Mas_Rapida,
    MAX(DATEDIFF(DAY, t_envio.Fecha, t_entrega.Fecha)) AS Entrega_Mas_Lenta
FROM Fact_Entregas fe
JOIN Dim_Proveedor dp ON fe.ID_Proveedor = dp.ID_Proveedor
JOIN Dim_Tiempo t_envio ON fe.Tiempo_Key_Envio = t_envio.Tiempo_Key
JOIN Dim_Tiempo t_entrega ON fe.Tiempo_Key_Entrega = t_entrega.Tiempo_Key
GROUP BY dp.NombreProveedor;
```

Conciliación Ventas vs. Entregas (Métricas de Pendientes): Para saber cuántas ventas se hicieron, cuánto se entregó y si hay alguna discrepancia en las cantidades.

```sql
SELECT
    fv.ID_Venta,
    dc.RazonSocial AS Cliente,
    fv.Cantidad AS Cantidad_Pedida,
    ISNULL(fe.CantidadProductos, 0) AS Cantidad_Entregada,
    (fv.Cantidad - ISNULL(fe.CantidadProductos, 0)) AS Diferencia
FROM Fact_Ventas fv
JOIN Dim_Cliente dc ON fv.ID_Cliente = dc.ID_Cliente
LEFT JOIN Fact_Entregas fe ON fv.ID_Venta = fe.ID_Venta
ORDER BY Diferencia DESC;
```

Vista rápida:

```sql
CREATE OR ALTER VIEW VW_Reporte_Logistico_Integral AS
SELECT
    fv.ID_Venta,
    dt.Descripcion AS Tienda,
    dc.RazonSocial AS Cliente,
    dp.Descripcion AS Producto,
    fv.Cantidad AS Cantidad_Vendida,
    (fv.PrecioVenta + fv.Total_IVA) AS Monto_Total_Venta,
    ISNULL(fe.CodigoEntrega, 'SIN ASIGNAR') AS CodigoEntrega,
    ISNULL(dal.NombreAlmacen, 'PENDIENTE') AS Origen,
    ISNULL(dprov.NombreProveedor, 'N/A') AS Transportista,
    ISNULL(dest.Descripcion_Estado, 'NO ENVIADO') AS Estado_Actual
FROM Fact_Ventas fv
JOIN Dim_Tienda dt ON fv.ID_Tienda = dt.ID_Tienda
JOIN Dim_Cliente dc ON fv.ID_Cliente = dc.ID_Cliente
JOIN Dim_Producto dp ON fv.ID_Producto = dp.ID_Producto
LEFT JOIN Fact_Entregas fe ON fv.ID_Venta = fe.ID_Venta
LEFT JOIN Dim_Almacen dal ON fe.ID_Almacen = dal.ID_Almacen
LEFT JOIN Dim_Proveedor dprov ON fe.ID_Proveedor = dprov.ID_Proveedor
LEFT JOIN Dim_EstadoPedido dest ON fe.ID_Estado = dest.ID_Estado;

SELECT * FROM VW_Reporte_Logistico_Integral;
```

Planteo del Dashboard

Para expandir el Dashboard y lograr un análisis de las nuevas Tablas incluidas en el DW se construyeron las siguientes páginas:

A. Tiempos Logísticos (Página 5):

Esta sección se agregó para monitorear la eficiencia operativa del flujo de entregas.

Gráficos de Barras (Tiempo Promedio por Proveedor/Almacén): Muestran el rendimiento de cada eslabón. Se utilizan para identificar cuellos de botella geográficos o de servicio.

KPIs de Control: Se muestran tarjetas con el Promedio de Días por Entrega y Días Máximos/Mínimos, permitiendo establecer expectativas reales para el cliente final.

B. Cumplimiento de SLA (Página 6):

El objetivo de esta página es medir la fiabilidad de la promesa de entrega.

Gráficos de Líneas (SLA % por Año/Mes): Muestran la tendencia de cumplimiento a lo largo del tiempo, permitiendo visualizar si la calidad del servicio mejora o empeora ante picos de demanda.

Matriz de Pedidos Retrasados por Proveedor: Una tabla de calor que cruza proveedores con años, diseñada para detectar rápidamente qué socios logísticos están fallando en el cumplimiento histórico.

C. Eficiencia y Riesgo Logístico (Página 7):

Se añadió para cruzar el rendimiento operativo con el impacto económico.

Gráfico de Barras Agrupadas (Costo vs. Entregas): Relaciona el Costo Promedio de Entrega con el volumen de pedidos. Su función es determinar la rentabilidad de cada proveedor según su costo y efectividad.

Gráfico de Dispersión/Líneas (Pedidos en Fecha): Monitorea la estabilidad de la operación mes a mes.

D. Calidad de Datos (Página 8):

Esta es la página de gobernanza del reporte, para asegurar que las decisiones se tomen sobre datos ciertos.

Tabla de Auditoría (Motivos de Rechazo): Identifica registros con errores lógicos, como el "CodEnt 10006 - Venta posterior a Entrega", lo cual indica fallas en el registro del sistema fuente que deben ser corregidas para no sesgar los KPIs de las otras páginas.

3. Otros Cambios

Se agregó al dashboard una estructura de navegación superior que facilita la navegación entre páginas y se incorporaron filtros pertinentes a cada página para facilitar el análisis.

4.Respondiendo las Preguntas de la Consigna

a) ¿Cuáles son los tiempos de entrega promedio por proveedor? Se pueden visualizar en la página 5 ("Tiempos Logísticos") en el gráfico "Tiempo Promedio de Entrega por Proveedor".

b) ¿Qué almacén maneja mejor sus tiempos de preparación y despacho? Esta pregunta la responde el gráfico "Tiempo Promedio de Entrega por Almacen" (Página 5)

c) ¿Qué porcentaje de pedidos llega en la fecha estimada? Este dato es el KPI principal de la página 5 .

d) ¿Cómo afecta la distancia de envío al costo total de entrega? Esta relación se puede visualizar en el grafico de dispersión “Distancia KM y Costo por entrega” de la pagina Eficiencia Y Riesgo Logistico (Pagina 7).

Desarrollo del Dashboard

Primero se trabajó con los datos originales, una vez que los gráficos necesarios estaban funcionando, y mostraban los mismos datos que lo que las querys en SQL arrojaban, se trabajó en la generación de los nuevos archivos CVS para ampliar el número de registros.

Se generaron con un Script de Python que está en la carpeta Generar Registros, cuando se corre, los archivos CVS se generan dentro de esa misma carpeta.

En la generación se incluyeron posibles errores de carga para simular fallos de carga del mundo real(Datos negativos en el precio, fechas de entregas anteriores a las de venta, ventas que no tienen entregas y fechas inexistentes).

También se respetaron que en los archivos originales datos como la distancia en KM no estuvieran. Pero para verificar que los gráficos los pudieran mostrar correctamente se insertaron mediante estas Querys datos en el DW:

```
UPDATE Fact_Ventas
SET CostoEntrega = ROUND((50 - 5) * RAND(CHECKSUM(NEWID())) + 5, 2)
WHERE CostoEntrega IS NULL OR CostoEntrega = 0;
```

```
UPDATE FE
SET FE.DistanciaKm = ROUND(
    (FV.PrecioVenta * 0.05)
    * (0.8 + (RAND(CHECKSUM(NEWID())) * 0.4)), 0)
FROM Fact_Entregas FE
INNER JOIN Fact_Ventas FV ON FE.ID_Venta = FV.ID_Venta;
```

Los Backgounds en la carpeta SVG Backgrounds se realizaron en Figma con la finalidad de facilitar el diseño y su consistencia en todas las paginas.

Conclusiones Finales

Durante el desarrollo el mayor desafío que encontré fue enfrentarme a la suciedad de los datos originales, pensar de que manera manejarlos y probar distintas formas en los SP. Investigué técnicas usuales en el ámbito profesional para manejar esto, de donde saque la idea de generar otra tabla para almacenar registros descartados, y luego fue mucha prueba y error, verificando paso a paso el resultado para saber que datos habían llegado a las tablas finales.

Las habilidades técnicas (duras) clave fueron el manejo de SQL para la orquestación y limpieza. Sin embargo, la habilidad blanda más importante que ejercite fue el pensamiento analítico, el poder separar un problema grande en muchos pequeños para avanzar paso a paso, y la atención al detalle para detectar esos fallos lógicos que a simple vista pasan desapercibidos.

La resiliencia para volver atrás, borrar y re-procesar las tablas de hechos y dimensiones hasta que los números en Power BI coincidieran exactamente con las consultas de SQL fue fundamental para el éxito del dashboard. Para no perderme en estos bucles use github para guardar las distintas versiones del programa cada vez que realizaba un cambio importante.

Durante este curso me alegro mucho poner en práctica los conocimientos básicos de diseño y de la herramienta Figma que había adquirido en un curso anterior de Diseño UI/UX, a pesar de que no pienso dedicarme a esa rama siento que me ayudó mucho para entender como transmitir mejor la información. Hacia adelante, percibo que necesito profundizar en la todavía más en el manejo de datos en SQL para poder procesarlos de manera más eficiente y limpia al pasarlos de una tabla a otra. Mi próximo paso será explorar herramientas de integración en la nube.

Para terminar, si un amigo me preguntara si vale la pena, le diría: "¡Hacelo sin dudar!". Aprender análisis de datos es como ganar superpoderes para entender cómo funciona el mundo real; pasás de ver simples archivos CSV a entender por qué un envío se demora o cómo mejorar la rentabilidad de un negocio. También es un ejercicio importante de pensamiento lógico y analítico que sirve para muchos aspectos de la vida. Recomendaría empezar por entender la lógica del modelado de datos (como Kimball), porque si la base está mal, toda la limpieza y gráficos posteriores servirán de poco.