Inteligencia de Negocios

Juan Nicolás García Esquibel

201717860

Laboratorio 3

Etapa 3. Proceso ETL

El proceso inicia con la extracción de archivos CSV desde Cloud Storage, donde se leen cuatro fuentes principales: StockItems, Orders, OrderLines y PackageTypes. Cada archivo ingresa con su esquema original y pasa inmediatamente a un flujo de transformación. Para StockItems, se aplica un Wrangler que estandariza columnas renombrando ciertos campos, eliminando atributos que no aportan y rellenando valores nulos para mantener consistencia. Esa tabla limpia se carga directamente como DimStockItem y también continúa hacia un join posterior con las líneas de pedido. Algo similar ocurre con los datos de fechas: a partir de Orders, un Wrangler extrae el OrderDate, genera DateID en formato YYYYMMDD y descompone ese ID en año, mes y día. Luego se hace una agregación para quedarse con un único registro por cada fecha, que después se carga como DimDates.

En paralelo, OrderLines se une con Orders para asociar cada línea con su fecha. Después, otro Wrangler ajusta el campo DateID eliminando el timestamp y convirtiéndolo a entero, ya que se usará como llave para unir con la tabla de fechas. Cuando las líneas ya están limpias y con su fecha normalizada, el flujo se une con StockItemsClean mediante StockItemID, incorporando precio unitario, tax rate y demás atributos del producto. Más adelante, el resultado se une con los PackageTypes, pero antes ese catálogo se prepara generando un UUID por cada tipo de paquete. Luego se filtran únicamente los tipos de empaque que aparecen como activos, aunque esto se hizo más por requerimientos prácticos del ejercicio: idealmente, ese join debería considerar la vigencia temporal (ValidFrom/ValidTo) para un manejo más completo del histórico tipo SCD2, pero Data Fusion tiene limitaciones y, por falta de experiencia en ese momento, no logré implementarlo así. Además, las columnas isActive y ValidTo las agregué externamente con un script porque, aunque ValidFrom era suficiente, me pareció más adecuado para un enfoque tipo 2 tener explícitos ambos campos. También intenté generarlos dentro de la herramienta, pero no encontré una solución directa ni con Wrangler ni con el plugin de JavaScript.

Tras ese join final, un Wrangler calcula el importe total por línea como Quantity × UnitPrice. Finalmente, los datos enriquecidos y con todas sus llaves foráneas se cargan en BigQuery como la tabla FactTableOrderLines. En paralelo, las tablas limpias de productos, fechas y tipos de empaque se cargan como dimensiones (DimStockItem, DimDates, DimPackageType). El resultado es un modelo tipo estrella donde la tabla de hechos queda completamente relacionada con sus dimensiones mediante llaves numéricas o UUIDs.

Como nota adicional, el diagrama original del enunciado planteaba que el join final recibiera tres fuentes simultáneamente, lo cual era incorrecto porque Data Fusion iba a intentar hacer el match de los mismos campos en las tres tablas al tiempo. Para evitar eso y asegurar que cada unión se hiciera con las columnas y llaves correspondientes, fue necesario separar el proceso en joins independientes: primero con StockItemsClean y luego con los PackageTypes procesados.

Etapa 4. Modelo multidimensional y análisis

4.1. Reconstrucción del modelo

La imagen se encuentra en el repositorio.

El modelo multidimensional propuesto sigue una estructura en estrella centrada en la tabla de hechos FactTableOrderLines, donde se registran las líneas de pedido con sus medidas principales como Quantity, PickedQuantity, TaxRate, UnitPrice, TotalPrice y el identificador degenerado OrderID. Esta tabla se relaciona con tres dimensiones: DimDates, que descompone DateID en atributos como año, mes, día y la fecha original; DimStockItem, que describe cada producto mediante atributos como nombre, marca, tamaño, tiempos de entrega y otros metadatos relevantes; y DimPackageType, modelada como una dimensión SCD tipo 2 que almacena cambios históricos en los atributos de empaque (nombre, color, tamaño y fechas de vigencia) mediante ValidFrom, ValidTo, isActive y un UUID que identifica cada versión.

4.2. Consultas SQL

MontlyOrdersData.sql

```sql
WITH OrderTotals AS (
  SELECT
    f.OrderID,
    d.Year,
    d.Month,
    SUM(f.TotalPrice) AS OrderTotal
  FROM
    `lab3-bi-20252.lab3_bi_conjunto.FactTableOrderLines` AS f
  JOIN
    `lab3-bi-20252.lab3_bi_conjunto.DimDates` AS d
  ON
    f.DateID = d.DateID
```

Se completó la consulta

Se está usando la cuota de procesamiento según demanda

Resultados de la consulta

Guardar los resultados | Abrir en

Información del trabajo | Resultados | Visualización | JSON | Detalles de la ejecución | Gráfico de ejecución

	Year	Month	OrdersCount	AvgOrderValue	TotalSales
1	2014	1	1791	41979.72663316...	75185690.39999...
2	2014	2	1538	41654.67308192...	64064887.19999...
3	2014	3	1586	44666.18909205...	70840575.89999...
4	2014	4	1739	43341.74611845...	75371296.5
5	2014	5	1908	44686.62358490...	85262077.80000...

Con estos resultados se puede analizar cómo evoluciona el volumen de órdenes y el valor promedio de cada una a lo largo del año, identificar meses con mayor o menor actividad y

entender si el crecimiento en ventas proviene de un aumento en la cantidad de órdenes, de un mayor valor por orden o de una combinación de ambos factores.

PickingRate.sql

```sql
SELECT
  si.StockItemID,
  si.StockItemName,
  SUM(f.Quantity)       AS OrderedQuantity,
  SUM(f.PickedQuantity) AS PickedQuantity,
  SAFE_DIVIDE(
    SUM(f.PickedQuantity),
    NULLIF(SUM(f.Quantity), 0)
  ) AS PickingRate
FROM
  `lab3-bi-20252.lab3_bi_conjunto.FactTableOrderLines` AS f
JOIN
```

Se completó la consulta

Se está usando la cuota de procesamiento según demanda

Resultados de la consulta

Guardar los resultados | Abrir en

Información del trabajo | Resultados | Visualización | JSON | Detalles de la ejecución | Gráfico de ejecución

Fila	StockItemID	StockItemName	OrderedQuantity	PickedQuantity	PickingRate
1	1217	Black and orange glass with car...	1984176	750384	0.378184193337...
2	1102	The Gu red shirt XML tag t-shirt ...	986256	373248	0.378449408672...
3	1101	The Gu red shirt XML tag t-shirt ...	1067256	414720	0.388585306618...
4	1110	The Gu red shirt XML tag t-shirt ...	1069200	431568	0.403636363636...
5	1228	Tape dispenser (Red)	920700	383400	0.416422287390...

En este caso es posible identificar qué productos presentan dificultades en el proceso de preparación de pedidos, comparando lo solicitado con lo efectivamente recogido. Al enfocarse únicamente en ítems con suficiente volumen, la consulta permite detectar aquellos cuyo desempeño operativo es más débil y que podrían estar generando retrasos o ineficiencias en el flujo logístico. Con esta información es más fácil priorizar revisiones en procesos internos, ajustar procedimientos de bodega o evaluar posibles problemas relacionados con inventario, ubicación o manipulación de ciertos productos.

StockItemsAnnualVolume.sql

```sql
SELECT
  d.Year,
  f.StockItemID,
  s.StockItemName,
  SUM(f.Quantity)        AS TotalUnits,
  AVG(f.UnitPrice)       AS AvgUnitPrice
FROM
  `lab3-bi-20252.lab3_bi_conjunto.FactTableOrderLines` AS f
JOIN
  `lab3-bi-20252.lab3_bi_conjunto.DimDates`      AS d
  ON f.DateID = d.DateID
JOIN
```

Se completó la consulta

Se está usando la cuota de procesamiento según demanda

Resultados de la consulta

Guardar los resultados | Abrir en

Información del trabajo | Resultados | Visualización | JSON | Detalles de la ejecución | Gráfico de ejecución

Fila	Year	StockItemID	StockItemName	TotalUnits	AvgUnitPrice
1	2013	1215	Black and orange fragile despat...	3135672	3.7
2	2013	1216	Black and orange fragile despat...	2805192	4.1
3	2013	1202	Shipping carton (Brown) 500x3...	2463750	2.55
4	2013	1222	Black and orange this way up d...	2313360	4.1
5	2013	1212	3 kg Courier post bag (White) 3...	2289600	0.66

La última query permite identificar los productos con mayor movimiento en cada año, mostrando cuáles alcanzan los volúmenes más altos y cómo varía su comportamiento a lo largo del tiempo. Al combinar la cantidad total de unidades vendidas con el precio promedio por unidad, se pueden detectar ítems con alta rotación, productos que tienden a ganar o perder relevancia y posibles tendencias en la valorización de ciertos artículos. El uso de un ranking por año facilita comparar períodos y enfocar el análisis en los ítems más representativos dentro de cada ciclo anual.

4.3. Preguntas

¿Qué diferencia existe entre una arquitectura *Data warehouse* y un *Data lakehouse*?

Una arquitectura de data warehouse se centra en consolidar datos principalmente estructurados, provenientes de distintos sistemas transaccionales, en un modelo bien definido (por ejemplo, esquemas en estrella o copo de nieve). La idea es tener una “versión única de la verdad” optimizada para consultas analíticas, reportes y tableros de control. Normalmente se trabaja con cargas por lotes, procesos ETL bien controlados y un fuerte énfasis en calidad, gobernanza y estabilidad del modelo. Es muy útil para indicadores clásicos de negocio, como ventas, inventarios o comportamiento de clientes, donde las preguntas están relativamente bien acotadas.

En cambio, una arquitectura de data lakehouse busca combinar lo mejor de un data lake y de un data warehouse. En lugar de limitarse a datos estructurados, permite almacenar también datos semiestructurados o no estructurados (logs, JSON, archivos de texto, incluso imágenes o señales de sensores) sobre formatos abiertos en almacenamiento barato, pero añadiendo capas que soportan esquemas, transacciones, control de versiones y rendimiento para consultas analíticas. Esto habilita escenarios donde conviven BI tradicional, exploración de datos y casos de machine learning sobre la misma base tecnológica, sin necesidad de mantener dos plataformas totalmente separadas.

¿Qué ventajas y desventajas observa al momento de implementar un ETL utilizando este tipo de herramientas respecto a desarrollarlo utilizando Python, Pandas y demás herramientas vistas durante la primera parte del curso?

El uso de herramientas como Data Fusion ofrece ventajas claras para procesos ETL en entornos de nube. La orquestación de tareas (extracción, transformación y carga), el manejo de errores, el despliegue y el monitoreo vienen bastante integrados, lo que reduce esfuerzo operativo y tiempos de configuración. La interfaz visual facilita que personas de diferentes perfiles entiendan el flujo de datos sin leer código, y la integración nativa con servicios como Storage y BigQuery hace más sencilla la construcción de pipelines productivos y repetibles.

Sin embargo, frente a un desarrollo con Python, Pandas y librerías afines, este tipo de herramientas puede quedarse corta en flexibilidad. Lógicas de negocio muy específicas, validaciones a la medida o integración con modelos de machine learning suelen ser más naturales de expresar y mantener en código que en bloques visuales. Además, el control de versiones, las pruebas automatizadas y las buenas prácticas de ingeniería de software suelen quedar mejor estructuradas en un repositorio que en un canvas gráfico, sobre todo en proyectos grandes o con varios desarrolladores.

En términos generales, las plataformas gestionadas como Data Fusion parecen más adecuadas cuando se buscan ETLs relativamente estables y estandarizados sobre fuentes bien definidas, aprovechando la infraestructura de nube. En cambio, un enfoque basado en Python y Pandas resulta más conveniente cuando se requiere alta flexibilidad, fuerte integración con otras piezas de software o análisis avanzado. En la formación recibida hasta ahora se ha trabajado principalmente con Python, por lo que es natural que exista cierta inclinación hacia este enfoque y que herramientas como Data Fusion impliquen una curva de aprendizaje considerable al inicio. Además, en un contexto real también habría que considerar el costo de mantener instancias de este tipo de servicios gestionados en GCP y evaluar si el presupuesto y la relación costo–beneficio justifican su uso frente a alternativas basadas en código. Aun así, con más conocimiento y experiencia, este tipo de plataformas puede convertirse en un complemento muy útil dependiendo del contexto y de las necesidades específicas del proyecto.

Explore Google Cloud Data Fusion y documente sus principales características, componentes y funcionalidades. Describa cómo se pueden crear, configurar y ejecutar flujos de datos dentro de la herramienta, y en qué escenarios resulta especialmente útil su uso.

Google Cloud Data Fusion es un servicio gestionado de integración de datos orientado a construir canalizaciones ETL/ELT sin necesidad de escribir código. Ofrece una interfaz drag and drop con numerosos conectores y transformaciones, integrada de forma nativa con Cloud Storage, Dataproc y BigQuery, además de funcionalidades de gobierno como linaje y metadatos centralizados. La herramienta se organiza alrededor de una instancia creada en GCP y de su interfaz (Cloud Data Fusion Studio), que incluye el Control Center, Pipeline Studio para diseñar flujos, Wrangler para perfilar datos y módulos de administración para configuraciones y monitoreo.

La construcción de pipelines se realiza en Pipeline Studio, donde se selecciona el tipo de flujo y se agregan nodos de tipo Source, Transform, Analytics y Sink, conectándolos visualmente según la lógica del proceso. Las transformaciones pueden realizarse mediante plugins estándar o apoyarse en Wrangler, y una vez configurado el pipeline, Data Fusion ejecuta la canalización y gestiona automáticamente los recursos necesarios. También permite programar ejecuciones, integrarse con eventos y revisar métricas y logs de cada trabajo.

Esta herramienta resulta especialmente útil cuando se requiere integrar múltiples fuentes heterogéneas hacia un data lake o un data warehouse, modernizar ETLs existentes y centralizar procesos de ingeniería de datos con buenas prácticas de gobierno. Los casos de uso destacan su utilidad al construir arquitecturas analíticas en la nube y unificar data marts, habilitando tanto cargas por lotes como escenarios casi en tiempo real. En general, Data Fusion reduce esfuerzo operativo y evita desarrollos ad hoc, permitiendo que el equipo se concentre en el diseño del modelo y la lógica de negocio más que en la infraestructura.

Tabla de hechos y medidas

En el modelo multidimensional construido, se identificó una tabla de hechos transaccional, ya que cada registro representa una línea individual de una orden dentro del sistema de ventas de Wide World Importers. El grano está definido como una sola línea de pedido y, por lo tanto, captura eventos puntuales y no acumulativos. La presencia de atributos como OrderLineID, OrderID (dimensión degenerada), StockItemID, DateID y PackageTypeID, junto con medidas asociadas directamente a cada transacción, confirma que se trata de un hecho de tipo transaccional y no de un snapshot periódico ni un snapshot acumulativo.

Las medidas presentes en la tabla de hechos corresponden principalmente a métricas aditivas, ya que pueden sumarse en todas las dimensiones del modelo. Entre ellas se encuentran Quantity, PickedQuantity, UnitPrice, TotalPrice y TaxRate. Quantity y PickedQuantity son medidas estrictamente aditivas; UnitPrice y TaxRate son medidas semiaditivas que suelen analizarse mediante promedios ponderados o expresiones derivadas; por su parte TotalPrice es una medida derivada calculada como Quantity multiplicado por UnitPrice. Estas medidas permiten realizar análisis por producto, fecha, tipo de empaque y orden, manteniendo consistencia con un modelo centrado en métricas de venta a nivel de línea de pedido.

Ajustes a la dimensión PackageTypes

Si la dimensión PackageTypes debe manejar historia de tipo cuatro, el modelo requiere separar la versión vigente de los registros históricos. Se mantiene una dimensión principal con un solo registro actual por PackageTypeID y una tabla de historial que almacena las versiones anteriores con sus fechas de vigencia y campos de control como ValidFrom y ValidTo. La dimensión actual conserva únicamente los atributos activos, mientras que la tabla histórica guarda los valores previos para permitir análisis temporales. En el ETL, se compara la información entrante contra la dimensión principal usando PackageTypeID como clave de negocio. Cuando se detecta un cambio en algún atributo, la versión previa se envía a la tabla histórica con su periodo cerrado y la dimensión principal se actualiza con los valores

nuevos, asegurando que la consulta operativa use siempre la versión vigente y que el historial completo quede centralizado en la tabla de historia.

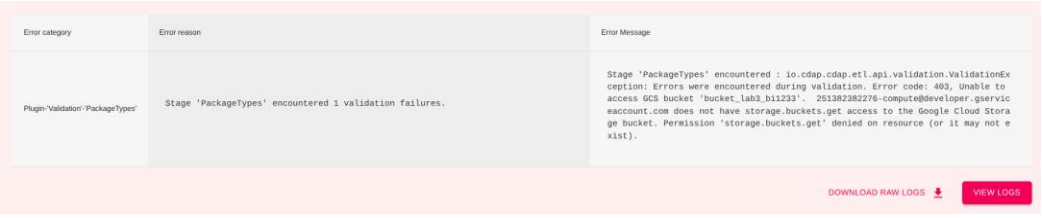
Exploración de Google Cloud Data Fusion

Google Cloud Data Fusion es un servicio totalmente gestionado para integrar datos en la nube y construir canalizaciones ETL y ELT sin escribir código. Ofrece una interfaz gráfica tipo drag and drop con numerosos conectores y transformaciones preconfiguradas, integrada con servicios como Cloud Storage, Dataproc y BigQuery, además de herramientas de gobierno como linaje de datos y metadatos centralizados. La plataforma se organiza alrededor de una instancia creada desde la consola de Google Cloud y de su interfaz principal, Cloud Data Fusion Studio, que incluye el Control Center, Pipeline Studio para diseñar flujos, Wrangler para perfilar y transformar datos de manera interactiva, y secciones administrativas para gestionar perfiles de cómputo, configuraciones y monitoreo.

La construcción de flujos se realiza principalmente en Pipeline Studio, donde se selecciona el tipo de pipeline y se agregan nodos de tipo Source, Transform, Analytics, Sink, condiciones y acciones, conectándolos visualmente según la lógica requerida. Las transformaciones pueden aplicarse mediante plugins o generarse desde Wrangler trabajando sobre una muestra de datos. Después de configurar parámetros como conexiones, esquemas o filtros, el pipeline se despliega y Data Fusion gestiona los recursos necesarios de forma automática. También permite programar ejecuciones, activarlas por eventos y consultar métricas o logs desde la vista detallada de la canalización.

Esta plataforma resulta especialmente útil cuando se necesitan integrar fuentes heterogéneas dentro de un data lake o un data warehouse, modernizar procesos ETL existentes y centralizar la ingeniería de datos con buenas prácticas de gobierno. La literatura y experiencias reales resaltan su utilidad para construir arquitecturas analíticas en la nube, unificar data marts y habilitar tanto cargas por lotes como procesos cercanos al tiempo real mediante conectores reutilizables. En este contexto, Data Fusion reduce trabajo operativo y evita desarrollos ad hoc, permitiendo que los equipos se concentren más en el diseño del modelo y la lógica de negocio que en la infraestructura.

¿Qué errores se le presentaron en el desarrollo del laboratorio y qué solución plantearon?

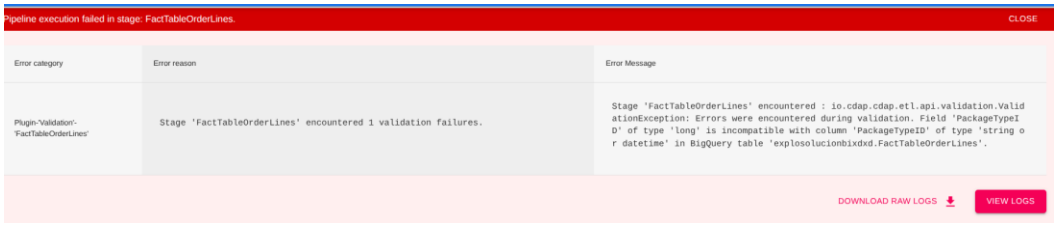


Uno de los primeros errores se presentó al importar el JSON proporcionado. Intenté ejecutar el pipeline de una vez, pero las referencias al bucket de GCS no eran válidas. La solución fue ingresar a cada uno de los stages de lectura desde Cloud Storage y actualizar el source para

apuntar al bucket que había creado en mi proyecto.

Description	Quantity	PickedQuant...	DateID	PackageTypeID
USB missile launcher (Green)	1	1	20130123	null
USB missile launcher (Green)	1	1	20140131	null
USB missile launcher (Green)	1	1	20140204	null
USB missile launcher (Green)	1	1	20150709	null
USB missile launcher (Green)	1	1	20150730	null
USB missile launcher (Green)	1	1	20150827	null
USB missile launcher (Green)	1	1	20151201	null
USB missile launcher (Green)	1	1	20160127	null
USB missile launcher (Green)	1	1	20160303	null
USB missile launcher (Green)	1	1	20160311	null
USB missile launcher (Green)	1	1	20130221	null
USB missile launcher (Green)	1	1	20130307	null

Luego, al explorar los datos generados por el pipeline, noté que en la tabla de hechos el campo PackageTypeID estaba quedando en nulo para la mayoría de registros. Al revisar el join, encontré que lo estaba construyendo de forma que exigía un match exacto entre las tres tablas involucradas, es decir, los identificadores de todas debían coincidir. Eso hacía que hubiera muy pocos registros coincidentes y, como el join era de tipo outer, se traía el resto de filas, pero con columnas como esta en nulo. Aunque en la descripción del ETL este procedimiento se explica con más detalle, finalmente se optó por dividir la lógica en dos joins separados,



Más adelante surgió un problema en la carga de la tabla de hechos, específicamente porque el campo PackageTypeID no coincidía con el tipo definido en el esquema final. Esto pasaba porque aún se estaba propagando un identificador incorrecto desde etapas intermedias del pipeline. La solución fue ajustar los joins y limpiar el esquema para asegurar que solo se utilizara el ID proveniente de la dimensión correspondiente.

Sin duda hubo otros errores adicionales, pero fueron menores y se resolvieron rápidamente, ajustando algún tipo de dato o eliminando tablas en BigQuery que habían quedado de ejecuciones previas.

Investigue cómo se puede conectar a través de Tableau o Power BI a su Big Query, donde están las tablas finales creadas y con datos. Documente los hallazgos e intente conectar los datos creados del laboratorio con una de estas herramientas para crear tableros de control.

Para conectar las tablas finales del laboratorio en BigQuery con Power BI, se revisó el conector nativo disponible en la versión web del servicio. Desde Get data → Google BigQuery, Power BI habilita una conexión cloud-to-cloud que permite acceder directamente al proyecto indicando el Billing Project ID y autenticándose con la cuenta de Google. Una vez establecida la conexión, es posible navegar por los datasets, seleccionar las tablas creadas por el ETL y cargarlas al modelo, ya sea en modo importación o mediante esquemas más cercanos a tiempo real según la configuración.

En general, la exploración mostró que el uso del conector nativo simplifica bastante la integración entre BigQuery y Power BI: la autenticación, la administración de consultas y el acceso a las tablas quedan resueltos por el propio servicio, lo que permite enfocarse en el diseño del modelo y de los tableros. Aun así, es importante considerar aspectos como el proyecto de facturación asociado, el modo de acceso (Import vs DirectQuery) y el impacto en costos y rendimiento al trabajar con conjuntos de datos alojados en BigQuery.