- ¿Qué diferencia existe entre una arquitectura *Data warehouse* y un *Data lakehouse*? ¿Qué tipo de arquitectura le recomendaría a WWI para complementar lo desarrollado en este laboratorio incluyendo fuentes de datos no estructuradas, análisis en tiempo real que incluyan simultáneamente tanto datos estructurados como no estructurados?

Un Data Warehouse es una arquitectura orientada a análisis estructurado, con esquemas definidos (estrella o copo de nieve), alta gobernanza y un enfoque en consultas analíticas sobre datos limpios e integrados. Por el contrario, un Data Lakehouse combina lo mejor de un data lake (capacidad de almacenar datos estructurados, semiestructurados y no estructurados en bruto) con las capacidades analíticas y de transacciones propias de un warehouse, permitiendo unificar procesamiento batch, streaming y análisis avanzado en un único sistema. Para WWI, cuya operación podría expandirse a análisis en tiempo real, integración de logs, archivos JSON, imágenes u otras fuentes no estructuradas, recomendaría evolucionar hacia un Data Lakehouse, ya que permite complementar el Data Warehouse del laboratorio con almacenamiento flexible, ejecución de machine learning y análisis simultáneos sobre datos estructurados y no estructurados, manteniendo gobernanza y consistencia.

- ¿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 visuales como Google Cloud Data Fusion ofrece ventajas importantes: reduce la complejidad del desarrollo, elimina la necesidad de programar pipelines desde cero, permite monitorear ejecuciones gráficamente y facilita la integración nativa con servicios de Google Cloud como Storage y BigQuery. Esto acelera el desarrollo y lo vuelve más comprensible para perfiles no técnicos. Sin embargo, también tiene desventajas: es menos flexible que programar directamente en Python/Pandas, limita el control detallado sobre transformaciones complejas, y depende del entorno administrado de GCP, lo cual puede generar costos y restricciones al personalizar lógica avanzada. Con Python se obtiene mayor libertad para modelar datos, optimizar procesos y aplicar expresiones complejas, pero a costa de mayor esfuerzo de implementación, despliegue y mantenimiento.

- 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.

Tanto Tableau como Power BI ofrecen conectores nativos para BigQuery, lo cual facilita la conexión directa a las tablas creadas en este laboratorio. En Power BI, basta con seleccionar *Obtener datos → Google BigQuery*, iniciar sesión con la cuenta de Google, elegir el proyecto y dataset, y luego cargar las tablas Dim y Fact directamente al modelo. En

Tableau, el proceso es similar mediante el conector “Google BigQuery”, donde se debe autenticar la cuenta, seleccionar proyecto, dataset y tablas, y luego construir visualizaciones con el esquema importado. En ambos casos, la conexión es estable y el rendimiento es alto gracias al procesamiento en BigQuery, permitiendo construir dashboards sin necesidad de exportar datos localmente. Para este laboratorio, la conexión funcionó correctamente, permitiendo visualizar las dimensiones y la tabla de hechos generadas por el ETL y construir tableros básicos de análisis.

- ¿Qué tipo de tablas de hechos y de medidas se identifican en el modelo multidimensional dado? justifique la respuests. Para la tabla de hechos indique si es factless, transaccional, periódica o snapshot acumulativo. Para las medidas indique su tipo (No aditiva, Semi-aditiva o Aditiva).

La tabla de hechos generada corresponde a una tabla transaccional, ya que cada registro representa el detalle de una línea de pedido (OrderLine), asociada a un momento específico y a una combinación puntual de dimensiones (producto, fecha, tipo de empaque, orden). No es una tabla factless, ni periódica, ni snapshot acumulativo, sino un registro granular de transacciones. Las medidas incluidas corresponden a: Quantity (aditiva), UnitPrice (semi-aditiva, porque no se suma sobre dimensiones como StockItem), PickedQuantity (aditiva), TaxRate (no aditiva) y TotalPrice (aditiva). Las medidas aditivas pueden agregarse en todas las dimensiones, mientras que las semi-aditivas solo son agregables en algunas, y las no aditivas requieren transformaciones especiales para análisis.

- Suponga que a la dimensión Package Types del modelo creado llega información nueva y se continua con el manejo de historia de sus atributos. ¿Qué ajustes a la dimensión Package Types y al proceso ETL se deben realizar para que, al cargar la información, se incluya un manejo de historia de atributos de dimensión tipo 4? Describa cómo queda la dimensión, los atributos afectados y el proceso a seguir.

Si la dimensión PackageTypes necesitara incorporar un manejo de historia SCD Tipo 4, sería necesario separar la dimensión en dos tablas: una dimensión principal con los atributos vigentes (DimPackageType) y una tabla histórica auxiliar (DimPackageTypeHistory) donde se almacenan todas las versiones pasadas de cada PackageType. El ETL debería modificarse para que, cuando llegue un registro nuevo cuyo PackageType ya exista, se copie la versión previa completa en la tabla histórica antes de actualizar los valores en la dimensión principal. De esta forma, DimPackageType contendría únicamente la versión actual de cada registro, mientras que DimPackageTypeHistory contendría todas las versiones desactualizadas, incluyendo columnas como StartDate, EndDate, IsCurrent y el surrogate key. Esta técnica facilita consultas históricas sin sobrecargar la dimensión principal.

- 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 una plataforma administrada para construir pipelines de integración de datos mediante una interfaz visual. Sus componentes principales incluyen: *Sources* (para leer datos de Storage, BigQuery, bases de datos y archivos), *Transformations* (Wrangler, Joiner, agregaciones, filtros), y *Sinks* (carga hacia BigQuery, GCS o bases externas). Entre sus funcionalidades destacan la creación de pipelines ETL sin necesidad de escribir código, el versionamiento y validación de flujos, la integración nativa con el ecosistema GCP, la transformación interactiva de datos con Wrangler, y la posibilidad de ejecutar trabajos de manera programada. Crear un flujo consiste en arrastrar nodos, configurar sus propiedades (schemas, recetas, joins) y finalmente desplegar el pipeline. Es especialmente útil en escenarios donde se requiere rapidez de desarrollo, integración con múltiples fuentes, limpieza de datos visual y despliegue en ambientes productivos sin manejar infraestructura.

- ¿Qué errores se le presentaron en el desarrollo del laboratorio y qué solución plantearon? Haga énfasis en los que fueron más difíciles de solucionar.

Durante el desarrollo del laboratorio surgieron varios errores, entre ellos dificultades con la visualización del esquema de entrada y salida en los nodos Wrangler, causado por conexiones incompletas o esquemas no propagados; esto se solucionó validando cada nodo y asegurando que todos los Sinks reconocieran el esquema del paso anterior. También se presentaron problemas al crear medidas como TotalPrice debido a tipos de datos inconsistentes, lo cual se corrigió ajustando los tipos en Wrangler antes de la multiplicación. Adicionalmente, aparecieron errores de permisos o lectura en BigQuery, solucionados habilitando la opción “Update Table Schema” en los Sinks. Finalmente, uno de los problemas más complejos fue la interpretación del manejo de historia para PackageTypes; se resolvió implementando un SCD Tipo 2 simplificado mediante columnas de control y carga incremental en vez de un proceso completo con joins históricos, manteniendo la coherencia del modelo sin complejidad adicional.