Documentación Proyecto Final

Data Engineer - Federico Almiron

Databricks Academy - Clear Tech

Índice

Índice de Figuras

Versión

Versión	Fecha (dd/mm/yyyy)	Autor	Descripción
1.0	26/10/2025	Federico Almiron	Documento inicial

Introducción

Este documento presenta el diseño técnico y la implementación de una solución de datos end-to-end desarrollada como parte del proyecto final de la Clear Tech Academy. El proyecto se centra en el dominio de ventas minoristas, utilizando un conjunto de datos públicos de transacciones de ventas de licores en el estado de Iowa.

El objetivo principal es emular un proyecto empresarial real, aplicando las metodologías y tecnologías de la plataforma Databricks para ingestar, procesar y modelar datos. La solución transforma datos crudos a nivel de línea de factura en un Data Warehouse (DWH) optimizado, capaz de responder preguntas de negocio críticas sobre rendimiento de ventas, rentabilidad y optimización de precios .

La implementación sigue la Arquitectura Medallion, utilizando Unity Catalog para la gobernanza de datos a través de tres esquemas:

1. Capa Bronze (NB1_Load_Raw_Data): Realiza la ingesta de los archivos CSV crudos desde un volumen. En esta etapa, se aplica un filtro de calidad de datos esencial, utilizando lógica try_cast para validar la estructura de las filas y excluir registros con columnas desplazadas o inconsistentes, asegurando que solo los datos válidos avancen al siguiente proceso.
2. Capa Silver (NB2_Load_Sales_Curated_Data): Toma los datos limpios de Bronze y aplica una serie de transformaciones robustas. Este proceso incluye la limpieza de valores cruciales nulos, la conversión estricta de tipos de datos para todas las columnas numéricas y de fecha, y el enriquecimiento de los datos mediante la creación de métricas calculadas clave, como total_cost, sale_margin y sale_cases. El resultado es una tabla (sales_curated) limpia, estandarizada y lista para el análisis.
3. Capa Gold (NB3_Load_Dimensions y NB4_Load_Fact_Sales): Construye el modelo dimensional (Modelo Estrella).
    - Primero, se cargan las dimensiones (dim_product, dim_seller, dim_store, dim_location) utilizando la sentencia MERGE para implementar una lógica de SCD Tipo 1 (actualización de cambios).
    - Finalmente, se carga la tabla de hechos (fact_sales) uniendo la tabla Silver con las dimensiones ya cargadas para obtener las llaves sustitutas. Se utiliza un MERGE en la llave natural (InvoiceLineNo) para cargar de forma incremental las transacciones.

El resultado final es un pipeline de datos automatizado y robusto que proporciona una fuente de datos en la capa Gold, optimizada para el análisis de inteligencia de negocio y la toma de decisiones.

Objetivo General

El objetivo principal es transformar el conjunto de datos transaccionales de "Iowa Liquor Sales" en un Data Warehouse (DWH) limpio, estructurado y listo para el análisis. Para esto, se implementa la arquitectura Medallion (Bronze, Silver y Gold) gobernada por Unity Catalog, culminando en un modelo estrella en la capa Gold. Esta solución final permite responder eficazmente a las preguntas de negocio clave sobre rendimiento de ventas, rentabilidad y optimización de precios.

Objetivos específicos

1. Implementar una arquitectura de datos Medallion sobre Databricks Unity Catalog, estableciendo los esquemas (Bronze, Silver, Gold) y el volumen de ingesta requeridos para gobernar el ciclo de vida de los datos.
2. Desarrollar un pipeline de ELT robusto para procesar los datos de ventas. Esto incluye la aplicación de reglas de calidad de datos en la capa Bronze para excluir registros inconsistentes, seguido de la limpieza, estandarización y enriquecimiento en la capa Silver.
3. Diseñar y poblar un Data Warehouse (DWH) en la capa Gold siguiendo un modelo estrella. Esto implica la creación y carga de tablas de dimensiones y una tabla de hechos centralizada.
4. Implementar una lógica de carga dimensional SCD Tipo 1 utilizando la sentencia MERGE.
5. Validar la solución analítica demostrando que el modelo de datos Gold puede responder eficazmente a las preguntas de negocio clave sobre tendencias temporales, rankings de ventas, rentabilidad y optimización de precios.

Diagrama de Arquitectura

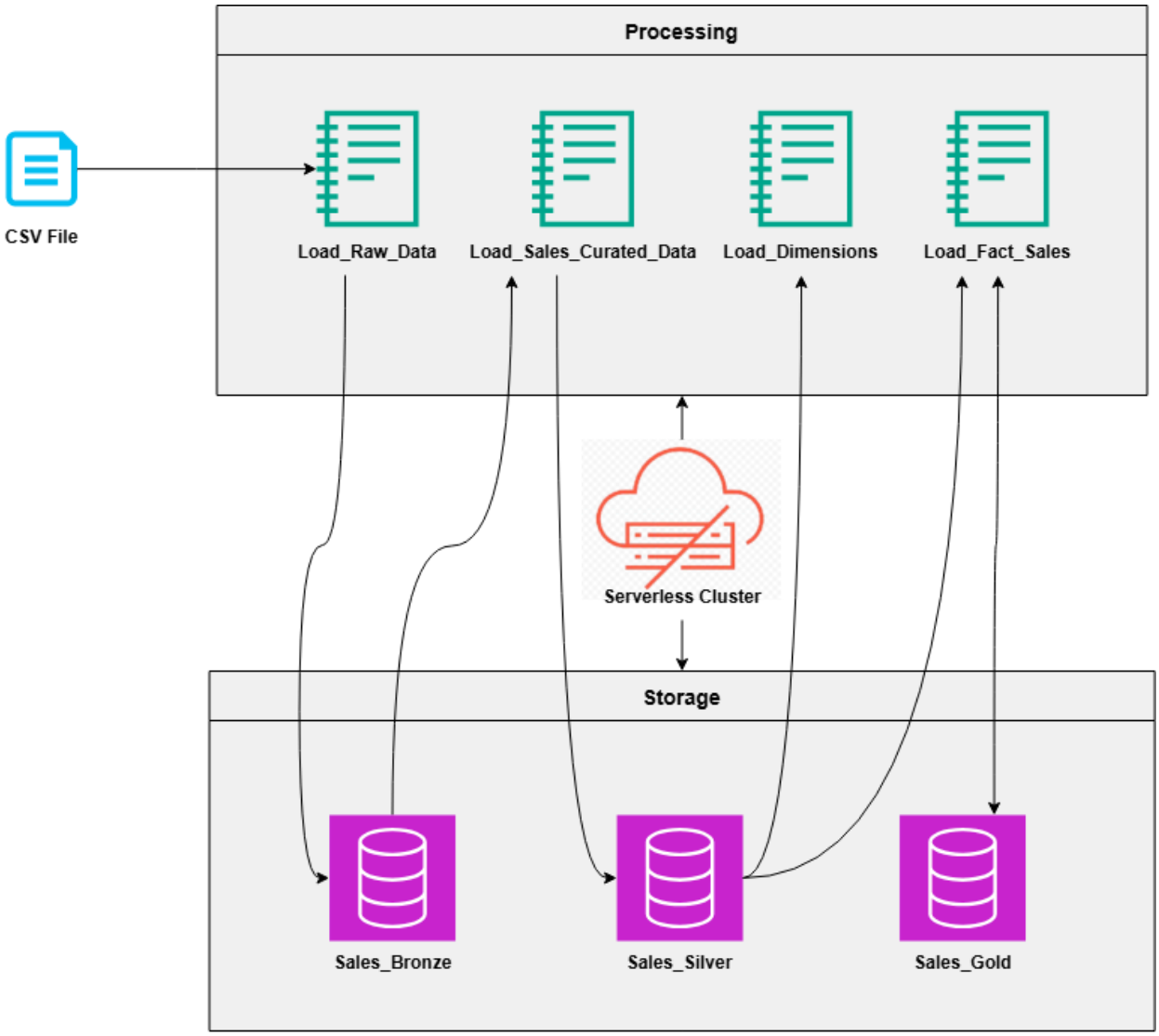

Figura 1: Diagrama de Arquitectura

Diagrama Lógico de la Solución

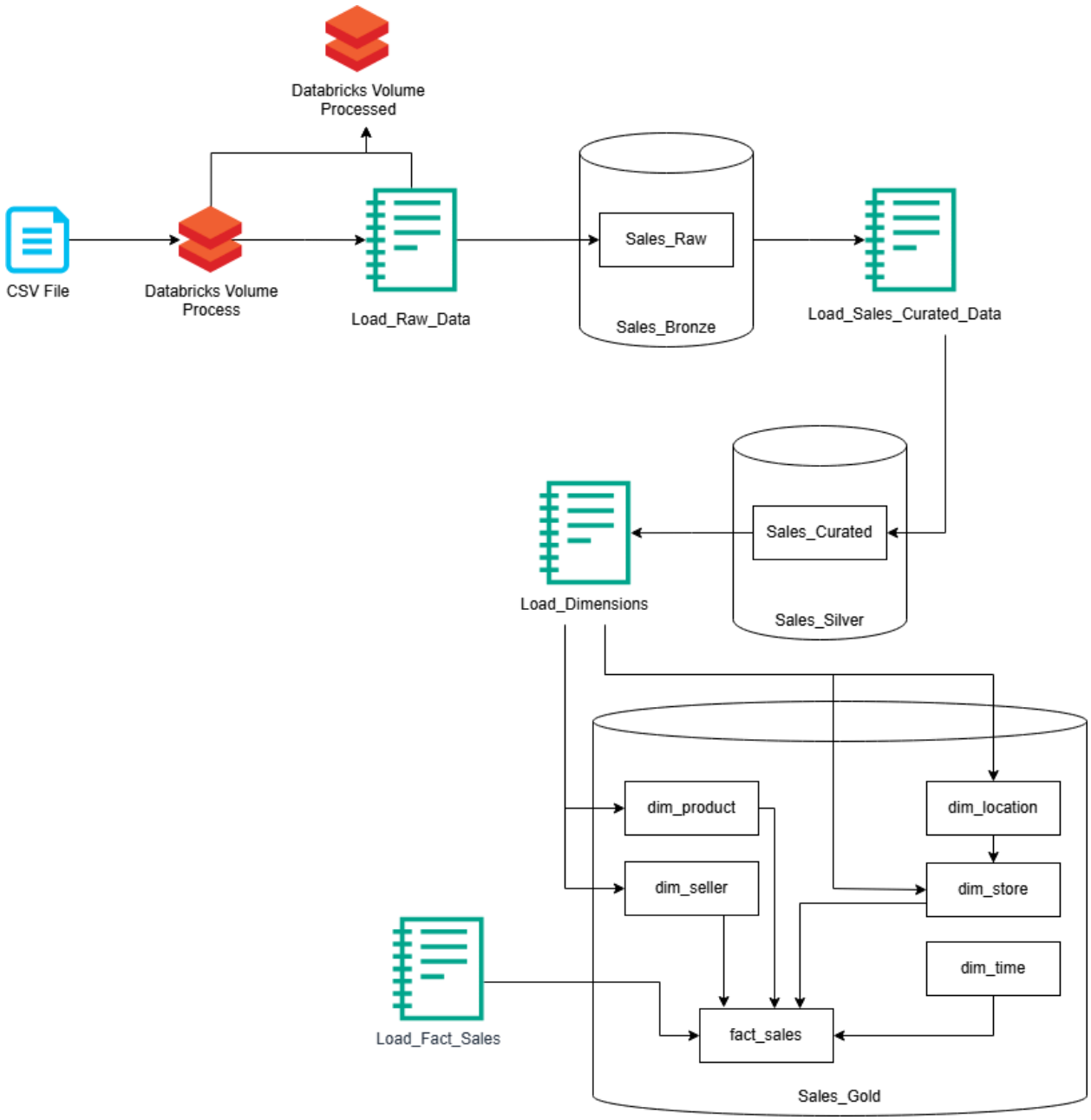

Figura 2: Diagrama lógico de la solución

Sección de Naming Convention

Archivos de entrada

La estructura de nombre de los archivos de entrada debe componerse por yyyymmdd-yyyymmdd.csv. Es decir, solo se admiten archivos csv que su nombre se encuentra conformado por dos fechas divididas por un underscore, la primera fecha hace referencia al registro más antiguo dentro de ese archivo y la segunda fecha al registro más reciente.

Notebooks

El nombre de los notebooks se encuentra conformado de forma inicial por los caracteres NB-N haciendo referencia al número de notebook que corresponde. El restante del nombre se compone por palabras descriptivas al respecto del notebook separadas por underscores.
El notebook perteneciente a la capa Bronze se denomina: “NB1_Load_Raw_Data”
El notebook perteneciente a la capa Silver se denomina: “NB2_Load_Sales_Curated_Data”
Los notebooks pertenecientes a la capa Gold se denominan: “NB3_Load_Dimensions” y “NB4_Load_Fact_Sales”.

Catálogo, Schemas y Tablas

El catálogo de la solución se denomina “sales”.
Los diferentes schemas se denominan: “sales_bronze”, “sales_silver” y “sales_gold”
La tabla perteneciente a la capa Bronze dentro del schema “sales_bronze” se denomina: “sales_raw”.
Dentro del schema “sales_bronze” se encuentra el volúmen “files”. Dentro de este volúmen residen dos carpetas llamadas “process” y “processed”.
La tabla perteneciente a la capa Silver dentro del schema “sales_silver” se denomina: “sales_curated”.
Dentro del schema “sales_gold” se encuentran las tablas que representan las dimensiones denominadas: “dim_store”, “dim_product”, “dim_location”, “dim_time” y “dim_seller”. Dentro de este schema también reside la tabla de hecho denominada “fact_sales”.

Jobs

La solución se compone de cuatro jobs. Uno para cada capa de procesamiento y otro para orquestar el procesamiento.
Los diferentes jobs se denominan: “WF_Sales_Main”, “WF_Load_Data_To_Bronze”, “WF_Load_Data_To_Silver” y “WF_Load_Data_To_Gold”.

Diseño Técnico de la Solución

Tecnologías utilizadas

Plataforma Databricks: Es una plataforma de análisis de datos unificada, basada en la nube y optimizada para Apache Spark. Proporciona un entorno colaborativo e integrado para la ingeniería de datos, la ciencia de datos y el machine learning.

Apache Spark (PySpark): Es un motor de procesamiento distribuido de código abierto diseñado para el análisis de datos a gran escala. Su API de Python, PySpark, permite a los desarrolladores utilizar la potencia del procesamiento distribuido de Spark con la sintaxis y las bibliotecas del lenguaje Python.

Delta Lake: Es un formato de almacenamiento en tablas de código abierto que se ejecuta sobre data lakes. Aporta transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), control de versiones (viaje en el tiempo) y gestión de metadatos, mejorando drásticamente la fiabilidad y el rendimiento de los data lakes estándar.

SQL (Spark SQL): Es un módulo de Apache Spark para el procesamiento de datos estructurados. Permite a los usuarios ejecutar consultas SQL estándar sobre grandes conjuntos de datos y DataFrames, integrando la potencia declarativa de SQL con el motor de procesamiento distribuido de Spark.

Componentes utilizados

Unity Catalog (UC): Es la solución de gobernanza unificada de datos dentro de la plataforma Databricks. Proporciona un control de acceso centralizado, auditoría, linaje de datos y capacidades de descubrimiento de datos a través de una jerarquía de objetos.

Databricks Notebooks: Una interfaz de usuario interactiva y basada en web que permite a los ingenieros y científicos de datos escribir y ejecutar código (en SQL, Python, Scala o R), visualizar resultados y compartir conocimientos en un solo documento. Son el principal entorno de desarrollo para los pipelines de ELT.

Databricks Volumes: Los volúmenes son áreas de almacenamiento persistente dentro del entorno Databricks, utilizadas para gestionar archivos de entrada, salidas intermedias y respaldos de procesamiento.
Funcionan como una extensión del sistema de archivos del workspace y permiten organizar los datos crudos y procesados de forma estructurada, facilitando el versionado y la recuperación de la información.

Databricks Jobs: Los jobs representan unidades de automatización dentro del ecosistema Databricks. Permiten programar y ejecutar notebooks, scripts o flujos de trabajo de forma

periódica o bajo demanda. Los jobs son esenciales para operacionalizar el pipeline de datos, ya que aseguran la ejecución ordenada de las tareas, el manejo de dependencias entre procesos y la captura de logs o errores durante su ejecución.

Triggers: Definen la periodicidad y condiciones de ejecución automática de los jobs. Permiten programar la ejecución del pipeline de datos en horarios determinados o ante eventos específicos, garantizando la actualización continua de la información en el Data Warehouse. Contribuyen a la automatización del flujo de datos y a la disponibilidad oportuna de los resultados analíticos.

DBFS (Databricks File System): Sistema de archivos distribuido propio de Databricks. Actúa como una capa intermedia entre el almacenamiento en la nube y el entorno de procesamiento, permitiendo acceder a los datos mediante rutas unificadas. Facilita la carga, lectura y organización de archivos desde notebooks o jobs, funcionando como punto de enlace entre el data lake y los procesos de análisis dentro de la plataforma.

Archivos requeridos

El procesamiento de la solución se basa en un conjunto de archivos CSV que contienen la información transaccional original del dataset “iowa_dataset”.
El sistema solo admite archivos con extensión .csv, asegurando así la homogeneidad en el formato de entrada y la compatibilidad con los métodos de lectura utilizados.
Con el objetivo de optimizar la ingesta y el control de carga el dataset original fue particionado en cuatro archivos independientes, manteniendo en todos ellos la misma estructura de columnas y formato de datos.
Cada archivo representa un segmento del dataset original y sigue una denominación estandarizada, que permite identificar fácilmente el rango de fechas o el período al que pertenece la información contenida.
El nombre de los archivos de entrada se encuentra conformado por dos fechas divididas por un underscore, la primera fecha hace referencia al registro más antiguo dentro de ese archivo y la segunda fecha al registro más reciente. (Ej: yyyymmdd-yyyymmdd.csv)
Los archivos deben encontrarse ubicados en el volumen de datos destinado a la ingesta (“files/process/”) antes de la ejecución del flujo de procesamiento correspondiente.
Una vez que los archivos son validados y procesados exitosamente, se aplica una política de archivado automático que consiste en el traslado de los mismos hacia un directorio histórico (“files /processed/”). Agregando al nombre original “_YYYYMMDD” que representa la fecha de procesamiento, siguiendo el formato año–mes–día.

Capas y objetos de procesamiento con su respectiva descripción y uso

El pipeline de datos del proyecto sigue la Arquitectura Medallion, organizando el procesamiento en tres capas distintas: Bronze, Silver y Gold.

Capa Bronze: Ingesta de Datos Crudos

La Capa Bronze, asociada al schema “sales_bronze”, actúa como la zona de aterrizaje para los datos crudos. Es gestionada por el notebook “NB1_Load_Raw_Data”. Este notebook primero valida los archivos CSV fuente en un volumen determinado(files/process/), asegurando que coincidan con un patrón de nombre esperado. Su lógica principal es aplicar un filtro de calidad de datos (valid_cond) que utiliza try_cast para identificar y descartar registros con columnas desplazadas o estructuralmente inconsistentes. Los datos validados se cargan en la tabla principal de esta capa, “sales_raw”. Esta tabla sirve como un archivo histórico de los datos fuente, una vez validados estructuralmente. Finalmente, el notebook mueve los archivos procesados a una carpeta de archivo (files/processed/), renombrándolos con la fecha del día de procesamiento.

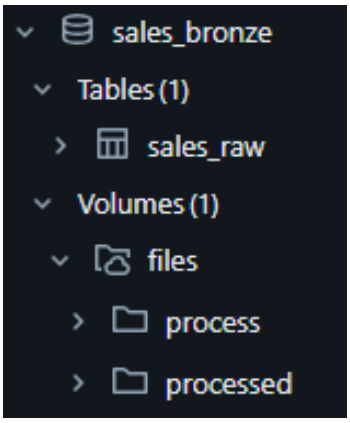

Figura 3: Objetos pertenecientes a la capa bronze.

Capa Silver: Limpieza y Enriquecimiento

La Capa Silver, que reside en el esquema “sales_silver”, transforma los datos de Bronze en un conjunto de datos curado, estandarizado y listo para el análisis. El propósito de esta capa(en el notebook “NB2_Load_Sales_Curated_Data”) es aplicar la lógica de negocio y las reglas de limpieza. Esto incluye la gestión de valores nulos (tanto eliminando registros como imputando valores), la conversión estricta de tipos de datos (como string a integer o decimal para cálculos precisos), y el enriquecimiento de los datos. Este enriquecimiento es clave, ya que aquí se calculan y añaden nuevas métricas de negocio que no existen en la fuente original.

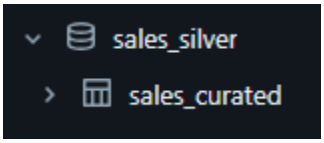

Figura 4: Objetos pertenecientes a la capa silver.

Capa Gold: Modelo Data Warehouse (DWH)

La Capa Gold, ubicada en el esquema “sales_gold”, consume los datos limpios de Silver y los reorganiza en un modelo optimizado para el análisis de inteligencia de negocio. En este proyecto, el propósito de la capa Gold es construir un Modelo Estrella. El proceso es manejado por los notebooks “NB3_Load_Dimensions” y “NB4_Load_Fact_Sales”. Los datos se desnormalizan en tablas de dimensiones y una tabla de hechos centralizada. Esta capa implementa la lógica de carga dimensional (SCD Tipo 1) mediante MERGE para manejar cambios en los atributos y resuelve las llaves foráneas para poblar la tabla de hechos. El resultado final es un DWH listo para ser consultado y responder a las preguntas de negocio.

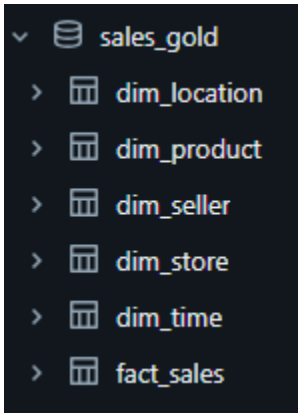

Figura 5: Objetos pertenecientes a la capa gold.

Facts

La solución de Data Warehouse se centra en una única tabla de hechos, “fact_sales”.
Esta tabla es el componente central del modelo estrella y está diseñada para capturar las métricas cuantitativas de cada evento de venta individual.

Propósito y Granularidad

El propósito de la tabla “fact_sales” es almacenar los detalles de rendimiento de cada línea de producto en una factura. Por lo tanto, la granularidad de la tabla es de una fila por línea de factura. Esta granularidad fina permite el análisis más detallado posible de las ventas, los costos y los márgenes a nivel de producto, tienda y día.

Estructura y Contenido

La tabla fact_sales contiene tres tipos principales de columnas:

1. Claves Sustitutas: Contiene las claves generadas por las tablas de dimensiones para conectarse con ellas. Estas son idProduct, idSeller, idDate, y idStore.
2. Claves de Negocio: La tabla preserva la clave natural de la transacción, InvoiceLineNo, que se utiliza como clave de unión para el MERGE. La clave primaria de la tabla, idSales, es una clave sustituta definida como identificador para garantizar la unicidad de cada registro en la tabla de hechos.
3. Métricas: Almacena todos los valores numéricos que se utilizarán en el análisis. Esto incluye métricas directamente de la fuente (como sale_bottles, state_bottle_cost, state_bottle_retail) y métricas enriquecidas que se calcularon en la capa Silver (como sale_dollars, total_cost, sale_margin, y sale_cases).

Carga de Datos

Los datos se cargan en “fact_sales” desde la tabla “sales_curated” (Silver) mediante el notebook “NB4_Load_Fact_Sales”. El proceso utiliza una sentencia MERGE que coincide en InvoiceLineNo. Esto permite que el pipeline actualice las transacciones existentes (WHEN MATCHED) e insertando nuevas (WHEN NOT MATCHED) en cada ejecución.

Dimensiones

En la capa Gold del modelo de datos se implementa una estructura de tipo estrella (Star Schema), compuesta por una tabla de hechos central (Fact_Sales) y un conjunto de tablas de dimensiones que describen los distintos ejes de análisis del negocio.
Las dimensiones permiten contextualizar las medidas numéricas almacenadas en la tabla de hechos, brindando una perspectiva descriptiva y categórica de los datos.

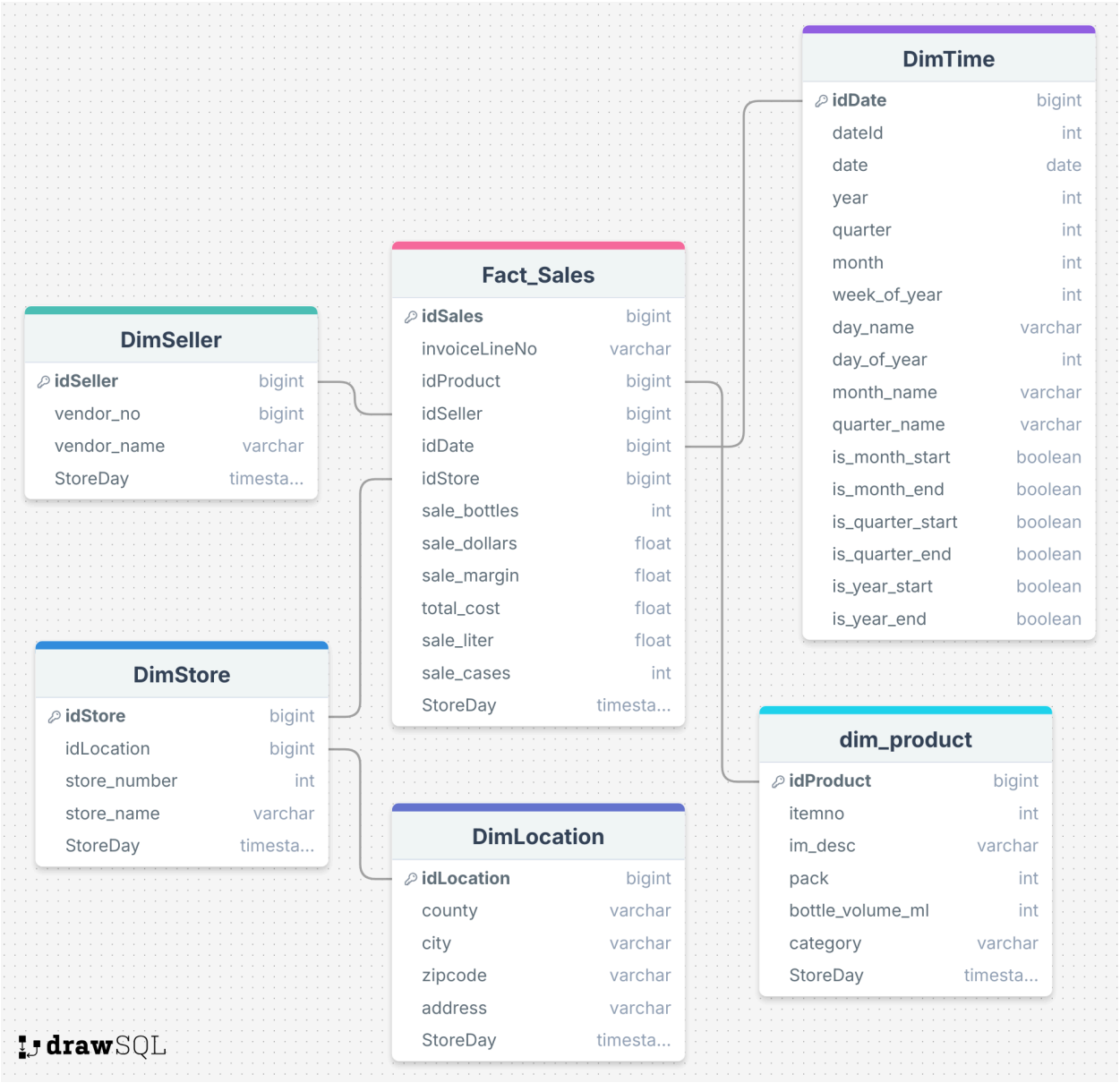

Figura 6: diagrama del datawarehouse

Dim_Product

La dimensión “Dim_Product” contiene información descriptiva sobre los productos comercializados.

Incluye atributos como el código del producto (itemno), su descripción (im_desc), cantidad de botellas por empaque (pack) y el volumen de cada unidad (bottle_volume_ml).

Esta dimensión permite analizar las ventas por tipo de producto o categoría, sirviendo como eje para el estudio del rendimiento de artículos específicos.

Dim_Seller

La dimensión “Dim_Seller” almacena información relativa a los proveedores o distribuidores de los productos.

Contiene atributos como el número de vendedor (vendor_no) y su nombre (vendor_name), proporcionando contexto para el análisis de ventas por proveedor, desempeño comercial y relación con el margen de rentabilidad.

Dim_Store

La dimensión “Dim_Store” representa las tiendas o puntos de venta donde se realizan las transacciones.

Incluye el número de tienda (store_number) y su nombre (store_name), funcionando como punto de referencia para evaluar el rendimiento de ventas por local o región geográfica.

Esta dimensión se vincula además con “Dim_Location” mediante una clave foránea que permite identificar la ubicación física de cada tienda.

Dim_Location

La dimensión “Dim_Location” describe la localización geográfica asociada a cada punto de venta.

Contiene información detallada del condado (county), ciudad (city), código postal (zipcode) y dirección (address).

Su propósito es habilitar análisis territoriales y comparativos entre distintas zonas geográficas, contribuyendo a la toma de decisiones estratégicas sobre distribución y expansión comercial.

Dim_Time

La dimensión “Dim_Time” provee el componente temporal del modelo, indispensable para la segmentación cronológica del análisis.

Incluye atributos derivados de la fecha como el año, trimestre, mes, semana y nombre del día, así como indicadores booleanos que identifican el inicio o fin de períodos.

Esta dimensión permite realizar comparaciones históricas, detectar tendencias estacionales y construir métricas de evolución temporal de las ventas.

Automatización

Para cumplir con los requerimientos de automatización, la solución se implementó utilizando Databricks Jobs. El diseño sigue un enfoque modular, compuesto por tres jobs independientes (uno para cada capa) y un cuarto job que actúa como orquestador principal.
Todos los jobs están parametrizados, permitiendo que variables como el catálogo y los nombres de esquema se pasen dinámicamente en tiempo de ejecución.
La solución se centra en un job orquestador principal, “WF_Sales_Main”, que define el flujo de ejecución de todo el proceso de ELT. Este orquestador invoca tres jobs de capa independientes en una secuencia estricta, asegurando el orden correcto del procesamiento: Load_Bronze → Load_Silver → Load_Gold. Esta dependencia garantiza que la capa Silver solo se procese si la ingesta de Bronze es exitosa, y la capa Gold solo se ejecute si la transformación de Silver finaliza correctamente, manteniendo así la integridad de los datos.

Figura 7: “WF_Sales_Main”

Cada job de capa es un flujo de trabajo autónomo. El job “WF_Load_Data_To_Bronze” ejecuta el notebook de ingesta “NB1_Load_Raw_Data”.
Toma como parámetros {catalog, bronze_table, bronze_schema}

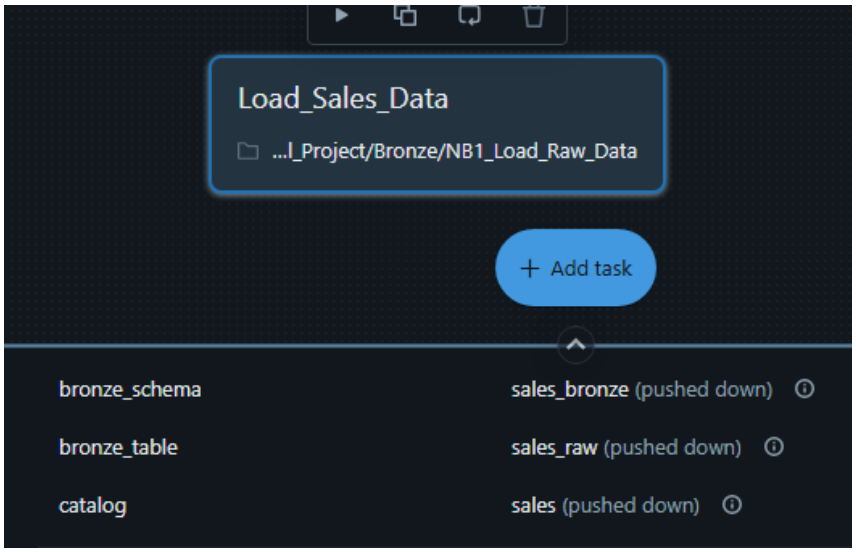

Figura 8: “WF_Load_Data_To_Bronze”

El job “WF_Load_Data_To_Silver” ejecuta el notebook de transformación “NB2_Load_Sales_Curated_Data”.
Toma como parámetros {catalog, bronze_table, bronze_schema, silver_schema, silver_table}

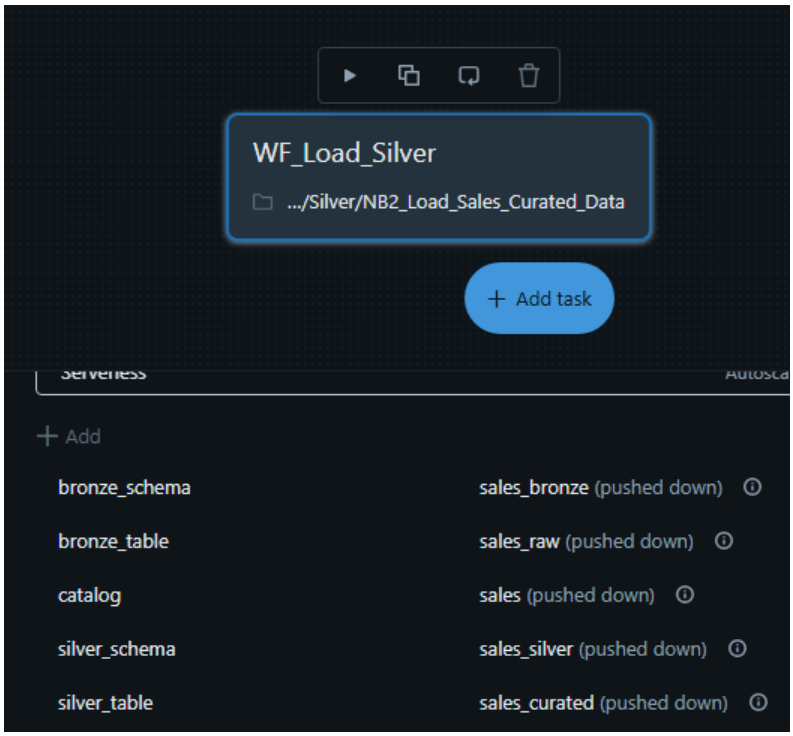

Figura 9: "WF_Load_Data_To_Silver"

Finalmente, el job "WF_Load_Data_To_Gold" es en sí mismo un flujo multi-tarea que primero ejecuta el notebook "NB3_Load_Dimensions" y, solo tras su finalización exitosa, procede a ejecutar "NB4_Load_Fact_Sales". Esta secuencia interna en el job Gold es crítica para asegurar la integridad referencial del modelo estrella, garantizando que las dimensiones existan antes de que la tabla de hechos intente referenciarlas.
Toma como parámetros {catalog, dim_location_table, dim_product_table, dim_seller_table, dim_store_table, dim_time_table, gold_fact_table, gold_schema, silver_schema, silver_table}

Figura 10: "WF_Load_Data_To_Gold"

Periodicidad de la Ejecución

La periodicidad de la ejecución se ha configurado directamente en el job orquestador principal, "WF_Sales_Main". Este job está configurado con un trigger de programación (schedule) que activa el pipeline completo. De acuerdo con los lineamientos del proyecto, la ejecución está programada para activarse dos veces al día, de lunes a viernes.

Adicionalmente, todos los jobs (incluyendo el orquestador principal y los jobs individuales de cada capa) están habilitados para su ejecución manual ("Run now"). Esto permite una flexibilidad total para realizar reprocesamientos, pruebas o ejecuciones de emergencia fuera del horario programado, en caso de ser necesario.

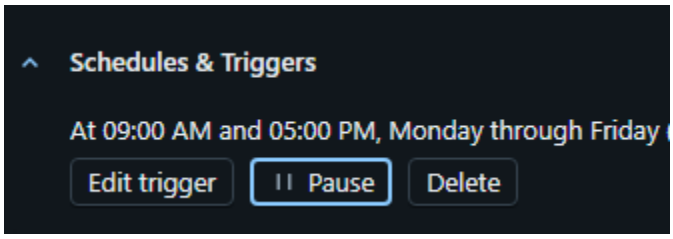

Figura 11: Trigger de procesamiento “WF_Sales_Main”

Manual de Uso

Esta guía describe el proceso para ejecutar el pipeline de ELT, desde la carga de archivos fuente hasta la consulta de los datos en la capa Gold.

Paso 1: Carga de Archivos Fuente
Para que el proceso de ingesta funcione, los archivos CSV fuente deben cargarse en la ubicación correcta.

1. Navegue al Catalog Explorer de Databricks.
2. Dentro del catálogo del proyecto (sales), expanda el esquema sales_bronze y seleccione el volumen files.
3. Cargue los archivos CSV de ventas en el directorio “process”. La ruta completa es: /Volumes/sales/sales_bronze/files/process/
4. Requisito de Nomenclatura: El notebook de ingesta está diseñado para buscar archivos .csv que coincidan con un patrón de nombre específico (ej. YYYYMMDD-YYYYMMDD_Sales.csv), como se define en la lógica de NB1_Load_Raw_Data.

Paso 2: Ejecución del Pipeline de ETL
El pipeline puede ejecutarse de dos maneras: mediante el orquestador principal (recomendado) o ejecutando el job de cada capa de forma independiente.
Opción A (Recomendada): Ejecución Orquestada
Este método ejecuta todo el pipeline de principio a fin y garantiza el orden correcto.

1. Navegue a la sección Jobs & Pipelines.
2. Busque y seleccione el job orquestador principal: “WF_Sales_Main”.
3. Haga clic en el botón "Run now" (Ejecutar ahora).
4. Este job ejecutará automáticamente las tres tareas en secuencia: Load_Bronze → Load_Silver → Load_Gold. Los parámetros del job (como los nombres de catálogo y esquemas) están preconfigurados, pero pueden ser modificados en la ejecución de ser necesario.

Opción B: Ejecución Manual por Capa
Este método es útil para reprocesar o depurar una capa específica.

1. Capa Bronze: Ejecuta el job “WF_Load_Data_To_Bronze”. Espere a que finalice exitosamente.
2. Capa Silver: Una vez completado el paso 1, ejecute el job “WF_Load_Data_To_Silver”.
3. Capa Gold: Una vez completado el paso 2, ejecute el job “WF_Load_Data_To_Gold”.

Paso 3: Verificación y Consulta de Resultados
Una vez que el job WF_Sales_Main (o WF_Load_Data_To_Gold) finalice exitosamente, los datos procesados estarán disponibles en el modelo estrella.

1. Navegue al Catalog Explorer y seleccione el esquema sales_gold.
2. Las tablas fact_sales, dim_product, dim_store, dim_location, dim_seller y dim_time estarán pobladas y listas para ser consultadas.

Nota sobre Reprocesamiento
El notebook de ingesta NB1_Load_Raw_Data está diseñado para que una vez un archivo en la carpeta “process” sea cargado exitosamente, es movido a la carpeta “processed” y se le añade un sufijo con la fecha de procesamiento. Para reprocesar el mismo archivo, este debe ser cargado nuevamente en la carpeta process.

Anexos

Notebooks

NB1_Load_Raw_Data.ipynb
NB2_Load_Sales_Curated_Data.ipynb
NB3_Load_Dimensions..ipynb
NB4_Load_Fact_Sales..ipynb

Queries

Preguntas de Negocio.ipynb