

Adventure Works Analytics

Proyecto de Inteligencia de Negocios



Autor:

Bryan Gustavo Guapulema Arellano

Plataforma:

Amazon Web Services

Tabla de contenido

1. Introducción	3
1.1. Descripción del documento	3
1.2. Criterios técnicos	3
1.3. Criterios de evaluación	3
1.4. Definiciones	3
1.4.1. Adventure Works Cycles:.....	3
1.4.2. Amazon Web Services (AWS)	4
1.4.3. Mejores prácticas aplicadas en el desarrollo	4
2. Infraestructura del proyecto.....	5
2.1. Fuentes de datos	5
2.2. Herramientas y servicios	6
2.3. Arquitectura	6
3. Metodología y Desarrollo del Proyecto	8
3.1. Creación del entorno y estructura del Data Lake (Amazon S3).....	8
3.2. Configuración de servicios de soporte	8
3.3. Implementación de roles e instancias Lambda	9
3.4. Ingesta de datos	9
3.5. ETL y volcado de datos.....	11
3.6. Configuración de carga incremental automatizada	11
3.7. Creación de vistas analíticas y conexión con Power BI	14
4. Visualización	15
4.1. Conexión a Power BI.....	15
4.2. Modelo de datos	16
4.3. Medidas DAX.....	16
5. Visualización	17
5.1. Dashboard.....	17
5.2. Historia construida a partir del dashboard	17
6. Mejores prácticas utilizadas.....	18
7. Recursos	20

1. Introducción

1.1. Descripción del documento

El presente documento tiene como objetivo describir de forma detallada el desarrollo técnico y analítico del proyecto de Inteligencia de Negocios “Adventure Works Analytics”, implementado en la plataforma **Amazon Web Services**.

Este informe recoge todos los procesos ejecutados para la creación de una solución end-to-end, que aborda el ciclo completo de vida de los datos desde la extracción y transformación de datos hasta la visualización final de indicadores comerciales. Adicionalmente se maneja carga incremental mes de los datos y un registro de logs relevantes del proceso automatizado.

El documento está estructurado para reflejar las fases del ciclo de vida del proyecto y los componentes utilizados en cada una de ellas. Además, se detallan las fuentes de información utilizadas, las herramientas de AWS empleadas, los modelos de datos creados, y el dashboard gerencial resultante.

1.2. Criterios técnicos

- Extracción de datos desde al menos dos fuentes distintas (APIs, archivos, bases de datos, etc.).
- Carga incremental por mes, el proceso deberá permitir la carga de varios meses de forma incremental mes a mes, se deberá programar o identificar método para que el proceso se ejecute sin intervención humana, es decir debe ser automático.
- Log errores, proceso deberá permitir identificar un los de errores
- Mejores prácticas, se deberá identificar las mejores prácticas para el desarrollo del proceso según la plataforma

1.3. Criterios de evaluación

- Extracción, transformación y carga correctamente ejecutadas en la plataforma elegida.
- Los procesos se ejecutan de forma diaria, además se pueden recargar datos históricos
- Cada fabricante según su plataforma contiene una serie de buenas prácticas en el desarrollo para explotar mejor su rendimiento a mejores costos
- Es fundamental que los procesos permitan identificar errores de carga, esto facilita el mantenimiento futuro de las soluciones
- Claridad en la explicación de resultados final.
- Soluciones novedosas, visualizaciones impactantes, enfoque original.

1.4. Definiciones

1.4.1. Adventure Works Cycles:

Compañía **manufacturera y distribuidora** de bicicletas de alto rendimiento, accesorios y componentes. La empresa opera a nivel internacional, con presencia en América del Norte, Europa y Asia, gestionando un modelo de negocio mixto que combina ventas al por mayor y distribución minorista a través de tiendas asociadas.

Su estructura organizativa incluye:

- Un equipo comercial dividido por territorios (país, región y ciudad).
- Un catálogo de productos organizado jerárquicamente en categorías y subcategorías.
- Relaciones estratégicas con proveedores (vendors) responsables del suministro de componentes y materias primas.

1.4.2. Amazon Web Services (AWS)

AWS es la plataforma de servicios en la nube más utilizada a nivel mundial. Proporciona infraestructura bajo demanda (IaaS), servicios de plataforma (PaaS) y herramientas analíticas y de inteligencia de negocios (BI). Permite crear arquitecturas altamente escalables, seguras y flexibles para procesamiento de datos, almacenamiento, integración y análisis.

1.4.3. Mejores prácticas aplicadas en el desarrollo

Durante el desarrollo del pipeline y el dashboard, se aplicaron varias buenas prácticas de ingeniería de datos en AWS, entre ellas:

1. Diseño modular y escalable

- Cada etapa (ingesta, transformación, carga y visualización) se implementó como funciones Lambda independientes.
- Las funciones se disparan mediante eventos S3, lo que permite escalar automáticamente cuando se suben nuevos archivos.

2. Organización de datos en capas

- Bronze Layer: Almacenó los archivos CSV originales de cada tabla fuente.
- Warehouse: Contuvo las tablas limpias y conformadas (transformaciones realizadas por).
- Views: Se diseñaron vistas de negocio (v_sales, v_dim_store, v_calendar_month_loaded) consumidas por Power BI.

3. Carga incremental segura

- Solo la tabla Orders se configuró como incremental, con un Lambda (h2-factsales-upsert-month) que detecta el mes (run_month) y añade nuevas filas sin duplicar meses previos.
- Se validaron escenarios de “mes vacío”, “mes repetido” y “múltiples meses simultáneos”.

4. Buenas prácticas de almacenamiento en S3

- Estructura clara y consistente:
- Separación por fuente y dominio, con nombres autoexplicativos y extensiones explícitas.

5. Registro y trazabilidad (Logging)

- Cada ejecución de Lambda guarda un registro en DynamoDB (tablas de control y error).
- Se habilitó CloudWatch Logs para auditoría y depuración.

6. Eficiencia de consultas en Athena

- Se usó formato Parquet comprimido (Snappy) para reducir costo y mejorar velocidad.
- Las consultas de transformación se ejecutaron con CTAS (CREATE TABLE AS SELECT).
- Las tablas externas se definieron de forma explícita (sin particiones innecesarias para este escenario).

7. Integración con herramientas BI

- Se conectó Power BI al catálogo hack2_aw_catalog de Athena mediante conexión Import (por rendimiento).
- Se programó un refresh manual del modelo tras nuevas cargas mensuales.

2. Infraestructura del proyecto

2.1. Fuentes de datos

El proyecto integra información proveniente de tres fuentes distintas, cada una con un propósito específico dentro del análisis comercial:

Fuente de datos	GitHub (CSV)
Tipo	Externa / Web
Descripción:	Archivos estructurados con información de clientes, empleados, órdenes, productos, subcategorías, categorías, proveedores y productos por proveedor.
Conexión	Disponible en: https://github.com/BryanGuapulema/AW_data_csv Obtenido de: https://github.com/valentinlog/adventureworksdataset
Contenido	<ul style="list-style-type: none"> - customers.csv - employees.csv - orders.csv - productcategories.csv - products.csv - productsubcategories.csv

Fuente de datos	MySQL (Stores)
Tipo	Base de datos relacional
Descripción:	Tabla relacional con información de tiendas físicas y empleados asociados.
Conexión	<ul style="list-style-type: none"> - hostname: relational.fel.cvut.cz - port: 3306 - username: guest - password: ctu-relational
Contenido	Tabla Stores

Fuente de datos	Excel (Presupuestos)
Tipo	Archivo local
Descripción:	Archivo con el presupuesto anual esperado de cada tienda.
Conexión	- Archivo local
Contenido	Hoja storesBudget

2.2. Herramientas y servicios

A. Lenguajes: Python (Lambdas), SQL (Athena), DAX (Power BI)

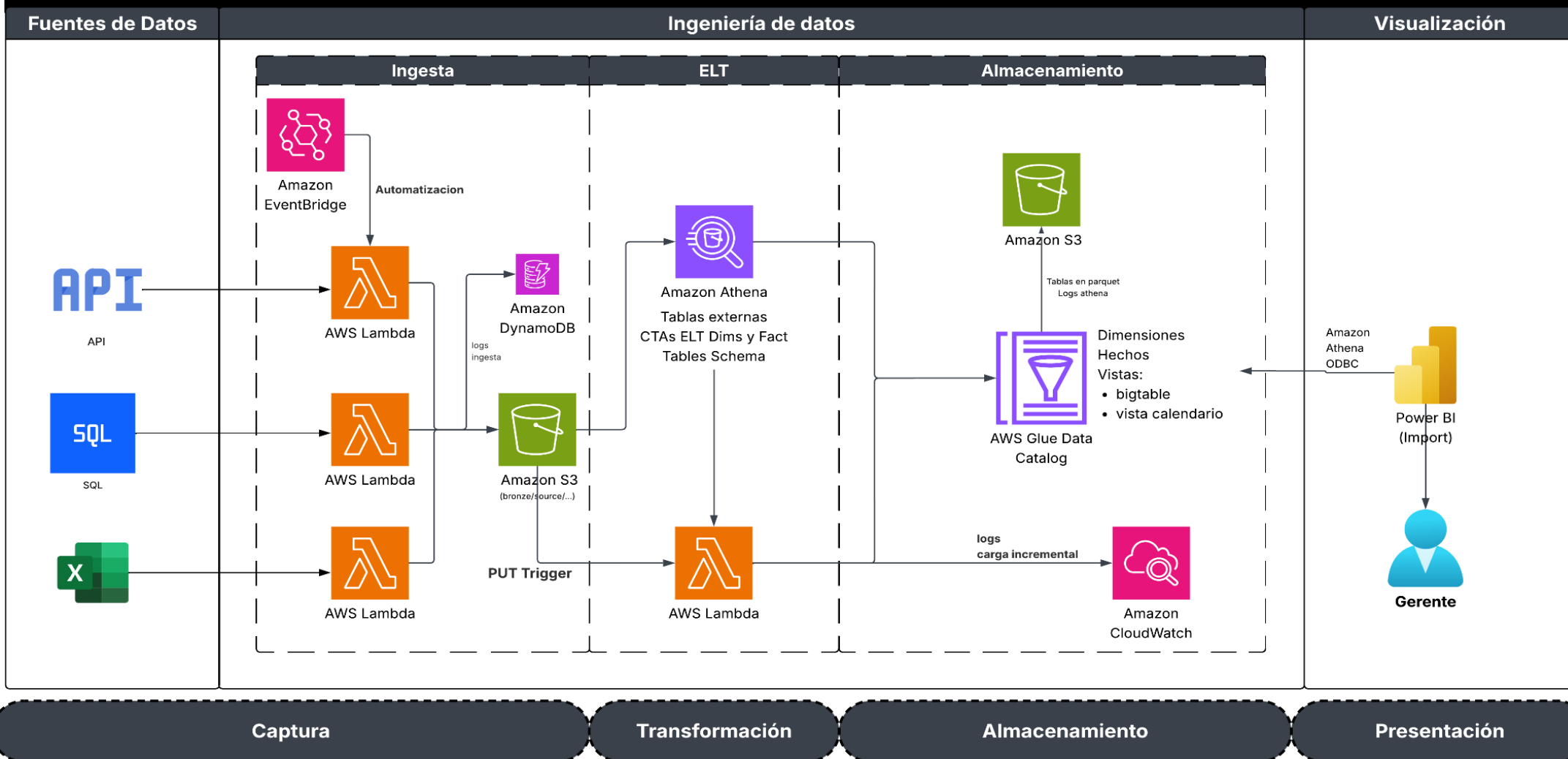
B. Amazon Web Services (AWS)

Herramienta / Servicio	Función	Función principal
Amazon S3	Almacenamiento	Data Lake en formato CSV estructurado por origen y tabla (bronze/source=.../table=...). Almacena las fuentes originales y los archivos del warehouse en formato parquet (incluye incrementales de ventas).
AWS Lambda	Transformación / Automatización	Ejecución del ELT para las tablas de dimensiones y la tabla de hechos fact_sales (con upsert mensual automático al cargar un nuevo archivo en S3).
Amazon Athena	Consulta y modelado	Motor SQL serverless para crear tablas externas, ejecutar transformaciones y generar vistas analíticas (v_sales, v_calendar_month_loaded, v_dim_store).
AWS Glue Data Catalog	Metadatos	Catálogo de bases y tablas (hack2_aw_catalog) que describe los esquemas para su uso en Athena.
Amazon DynamoDB	Monitoreo / Control	Registro de logs de ejecución y errores por tabla y mes (CONTROL_TABLE, ERROR_TABLE) para auditoría y mantenimiento.
Amazon CloudWatch	Observabilidad	Captura de logs de Lambda, errores y métricas de ejecución para diagnóstico operativo.
Amazon EventBridge	Orquestación (Programación diaria)	Servicio planificado para disparar diariamente la Lambda de ingesta o verificación de nuevos archivos de Orders. <i>(pendiente de implementación)</i>
Amazon IAM	Seguridad y autenticación	Gestión de roles y políticas de acceso mínimo entre S3, Lambda y Athena.
Power BI Desktop	Visualización	Conexión en modo Import a las vistas de Athena. Cálculos DAX para ventas acumuladas, brechas y porcentajes de cumplimiento de presupuesto.

2.3. Arquitectura



Amazon Web Services



3. Metodología y Desarrollo del Proyecto

3.1. Creación del entorno y estructura del Data Lake (Amazon S3)

El proyecto se inició con la configuración del entorno en Amazon S3, que sirvió como Data Lake central del proceso. Así pues, se creó el bucket bg-hack2-aw-datalake2, activando el cifrado por defecto (SSE-S3) y bloqueando el acceso público para garantizar la seguridad de los datos.

Dentro del bucket se organizó la siguiente estructura de carpetas, diseñada para soportar las distintas etapas del proceso:

- **bronze/** → Contiene los archivos fuente originales:
 - **source_metadata/**: archivo de configuración sources.json con la definición de todas las fuentes y sus parámetros. (véase el archivo en el repositorio de GitHub: <https://github.com/BryanGuapulema/Hackaton-2/blob/main/sources.json>)
 - **source=github/**: tablas CSV descargadas de GitHub.
 - **source=mysql/**: extracción de datos desde MySQL.
 - **source=excel/**: almacenamiento de presupuestos (Budget) provenientes de Excel.
- **warehouse/** → Carpeta donde se almacenan los resultados finales del procesamiento (archivos Parquet) generados al insertar los datos en las tablas del catálogo. Esta carpeta corresponde al producto final del proceso ELT.
- **logs/** → Contiene los resultados de las consultas ejecutadas en Athena (logs/athena-results/).

Esta estructura aseguró un flujo ordenado desde la ingesta de datos hasta su modelado final, manteniendo el historial de archivos por mes de carga.

3.2. Configuración de servicios de soporte

Se implementaron varios servicios de soporte en AWS para garantizar la seguridad, trazabilidad y operatividad del flujo.

- **AWS Secrets Manager**: Se almacenaron de manera segura las credenciales de acceso al origen MySQL utilizado para la tabla *Stores*. El secreto se registró bajo el nombre `hack2/mysql/stores`, con los campos: *username*, *password*, *host*, *port* y *database*. Este secreto fue consumido posteriormente por la función Lambda `ingest_mysql_stores`.
- **Amazon DynamoDB**: Se crearon dos tablas que permiten el seguimiento y control de las ejecuciones:
 - **ELT_control**: registra cada ejecución las fases de ingesta con atributos como `run_id`, `run_month`, `table`, `source`, `status`, `records_out`, `started_at` y `ended_at`.
 - **ELT_error_logs**: almacena los errores detectados durante las fases de ingesta incluyendo `error_code`, `message` y `severity`.

Ambas tablas se configuraron con modo On-demand para optimizar costos y permitir un registro ilimitado sin configuración de capacidad.

- **AWS Glue Data Catalog:** Se creó la base `hack2_aw_catalog`, utilizada para registrar las tablas externas (`ext_`) y las tablas finales (`dim_`, `fact_*`). Este catálogo permitió la gestión centralizada de metadatos accesibles desde Athena y Power BI.
- **Amazon Athena:** Se configuró como el motor principal de consultas SQL y de transformación de datos. En la sección Settings, se estableció la ruta de resultados: `s3://bg-hack2-aw-datalake2/logs/athena-results/`. De esta forma, cada consulta ejecutada en Athena genera un archivo de salida y su correspondiente registro.

3.3. Implementación de roles e instancias Lambda

Para garantizar el principio de separación de responsabilidades, se crearon tres roles IAM con permisos específicos:

- **lambda-ingest-role:** Incluye permisos de acceso total a S3, DynamoDB, CloudWatch y lectura en Secrets Manager. Se utiliza en las funciones de ingesta de datos.
- **lambda-transform-role:** Con permisos sobre S3, DynamoDB, Glue y Athena, utilizado para funciones de transformación y carga de datos procesados.
- **lambda-upsert-role:** Rol específico para ejecutar el proceso incremental de ventas mensuales, con permisos sobre Athena, Glue y S3.









3.4. Ingesta de datos

Para la ingesta automatizada de los distintos orígenes de datos se implementaron las siguientes funciones Lambda:

- **ingest_csv_github:** Descarga automáticamente los archivos CSV desde GitHub según la configuración del `sources.json`. Para la tabla *Orders*, filtra los registros por mes (`run_month`) en función del campo *OrderDate*, generando archivos como `orders_YYYY-MM.csv` en la ruta: `bronze/source=github/table=orders/`

≡ [Amazon S3](#) > [Buckets](#) > [bg-hack2-aw-datalake2](#) > [bronze/](#) > [source=github/](#) > [table=orders/](#)

Objetos (19)


 Copiar URI de S3
  Copiar URL
  Descargar
  Abrir
  Eliminar
  Acciones
  Crea

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

🔍

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	orders_2012-08.csv	csv	16 Oct 2025 12:25:46 PM -05	168.9 KB	Estándar
<input type="checkbox"/>	orders_2012-09.csv	csv	16 Oct 2025 12:26:14 PM -05	284.5 KB	Estándar
<input type="checkbox"/>	orders_2012-10.csv	csv	16 Oct 2025 12:26:42 PM -05	213.0 KB	Estándar
<input type="checkbox"/>	orders_2012-11.csv	csv	16 Oct 2025 12:27:10 PM -05	119.5 KB	Estándar







- **ingest_mysql_stores:** Extrae los datos desde la base MySQL (tabla Stores) utilizando las credenciales del Secrets Manager. El resultado se guarda en formato CSV en: `bronze/source=mysql/table=stores/csv/`

Amazon S3 > Buckets > bg-hack2-aw-datalake2 > bronze/ > source=mysql/ > table=stores/

table=stores/


Objetos Propiedades

Objetos (1)


 Copiar URI de S3
  Copiar URL
  Descargar
  Abrir
  Eliminar
 Acciones

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de los objetos. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	 stores.csv	csv	15 Oct 2025 7:11:25 AM -05	21.0 KB	Estándar








- **ingest_excel_storesBudget:** Lee el archivo de presupuesto en formato Excel desde la carpeta y lo almacena en `bronze/source=excel/table=storesBudget/`

Amazon S3 > Buckets > bg-hack2-aw-datalake2 > bronze/ > source=excel/ > table=storesBudget/ > csv/

csv/


Objetos Propiedades

Objetos (1)


 Copiar URI de S3
  Copiar URL
  Descargar
  Abrir
  Eliminar
 Acciones
  Crear

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de los objetos. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	 storesBudget.csv	csv	15 Oct 2025 9:38:14 AM -05	11.9 KB	Estándar

Nótese que tras la ejecución de cada uno de estas funciones lambda se registra el número de registros procesados en las tablas de DynamoDB.

DynamoDB > Explorar elementos > etl_control

Completado - Elementos devueltos: 10 Elementos escaneados: 10 - Eficiencia: 100% - RCU consumidas: 2

Tabla: etl_control: elementos devueltos (10)

El análisis inició el octubre 16, 2025, 18:17:24

<input type="checkbox"/>	run_id (Cadena)	ended_at	note	records_out	run_month	source
<input type="checkbox"/>	csv_2012-06_2012-07...	2025-10-16...	wrote bron...	983	2012-11	github
<input type="checkbox"/>	csv_2012-04_2012-05...	2025-10-16...	wrote bron...	1723	2012-05	github
<input type="checkbox"/>	csv_2012-01_2012-02...	2025-10-16...	wrote bron...	1112	2012-03	github
<input type="checkbox"/>	csv_2011-10_2011-11...	2025-10-15...	wrote bron...	377	2011-12	github

El código usado en cada uno de las funciones lambda se encuentra disponible en :

https://github.com/BryanGuapulema/Hackaton-2/tree/main/lambda_ingest

3.5. ETL y volcado de datos

Para la transformación de los datos de cada una de las tablas se optó por un proceso ELT (extracción, carga y transformación) cuya transformación se ejecuta en la base de datos. Esta fase siguió las siguientes etapas:

1. **Creación de tablas externas:** Se definieron las tablas externas (ext_customers, ext_employee, ext_products, ext_orders, etc.) en Athena apuntando a los archivos CSV almacenados en Bronze.

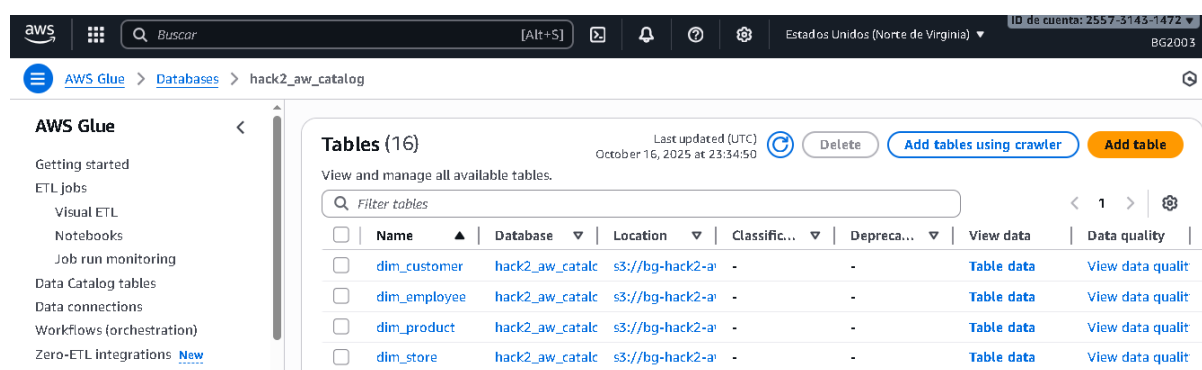
El código usado se encuentra disponible en:
https://github.com/BryanGuapulema/Hackaton-2/tree/main/athena_etl_warehouse/1_tablas_externas

2. **Creación de tablas en el catálogo:** Se creó la estructura de las tablas de destino en Glue Data Catalog (dim_customer, dim_employee, dim_product, dim_store, fact_sales), todas dentro de la base hack2_aw_catalog.

El código usado se encuentra disponible en:
https://github.com/BryanGuapulema/Hackaton-2/tree/main/athena_etl_warehouse/2_estructura_warehouse

3. **Volcado de datos en Glue Data Catalog** Mediante sentencias SQL ejecutadas en Athena se transfirieron los datos desde las tablas externas hacia las tablas definitivas del catálogo aplicando CTAS. Adicionalmente, los resultados se almacenaron en formato Parquet dentro de la carpeta: s3://bg-hack2-aw-datalake2/warehouse/

El código usado se encuentra disponible en:
https://github.com/BryanGuapulema/Hackaton-2/tree/main/athena_etl_warehouse/3_insert_dims_warehouse



Name	Database	Location	Classification	Deprecation	View data	Data quality
dim_customer	hack2_aw_catalog	s3://bg-hack2-aw-datalake2/warehouse/	-	-	Table data	View data quality
dim_employee	hack2_aw_catalog	s3://bg-hack2-aw-datalake2/warehouse/	-	-	Table data	View data quality
dim_product	hack2_aw_catalog	s3://bg-hack2-aw-datalake2/warehouse/	-	-	Table data	View data quality
dim_store	hack2_aw_catalog	s3://bg-hack2-aw-datalake2/warehouse/	-	-	Table data	View data quality

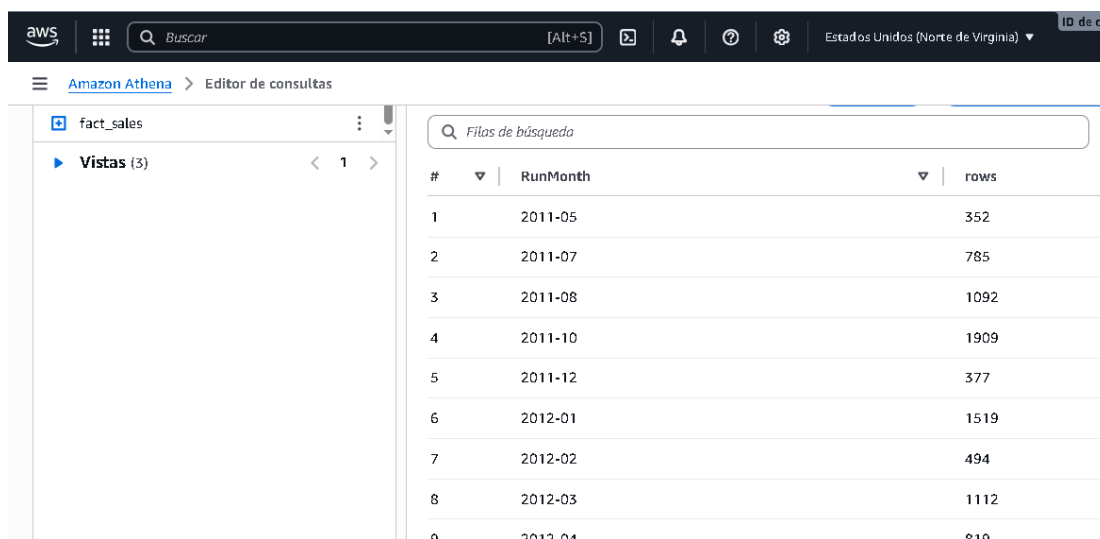
3.6. Configuración de carga incremental automatizada

1. **Automatización incremental de la tabla de hechos (fact_sales):**
 Una vez cargadas las dimensiones, se implementó el proceso de carga incremental con una función lambda (h2-factsales-upsert-month) la cual:

- Detecta el archivo orders_YYYY-MM.csv agregado.
- Identifica el mes correspondiente (run_month).

- Ejecuta un proceso INSERT INTO fact_sales para añadir únicamente los nuevos registros del mes.
- Registra la ejecución y resultados en DynamoDB (status, cantidad de registros, tiempo).

El código usado se encuentra disponible en:
<https://github.com/BryanGuapulema/Hackaton-2/tree/main/h2-factsales-upsert-month>



The screenshot shows the Amazon Athena console interface. On the left, there's a sidebar with 'fact_sales' and 'Vistas (3)'. The main area displays a table with the following data:

#	RunMonth	rows
1	2011-05	352
2	2011-07	785
3	2011-08	1092
4	2011-10	1909
5	2011-12	377
6	2012-01	1519
7	2012-02	494
8	2012-03	1112
9	2012-04	819

- 2. Trigger para la carga y transformación automática:** Para garantizar la automatización del proceso se configuró un trigger S3 que invoca automáticamente la Lambda h2-factsales-upsert-month cada vez que se crea un nuevo archivo .csv dentro de bronze/source=github/table=orders/

Esto asegura que el proceso incremental se ejecute inmediatamente al ingresar un nuevo mes de datos.

- 3. Automatización de ejecución diaria:**

Para la ejecución continua de este proceso se creó una función lambda (h2-scheduler-next-month) que busca en **DynamoDB la tabla etl_control** (DynamoDB) el **último run_month exitoso** de orders, calcula el **mes siguiente** y si este **no supera 2014-05**, invoca ingest_csv_github con:

```
{"run_months": ["YYYY-MM"]}
```

El código usado se encuentra disponible en:
https://github.com/BryanGuapulema/Hackaton-2/tree/main/lambda_daily_scheduler

Adicionalmente, se creó una regla en Amazon EventBridge con expresión cron que ejecute la función lambda h2-scheduler-next-month de forma diaria, cargando automáticamente un nuevo archivo mensual.

Amazon EventBridge > Reglas > Crear regla

Amazon EventBridge

Panel

▼ Recursos para desarrolladores

Obtenga información

Entorno aislado

Quick Starts

▼ Autobuses

Buses de eventos [Actualizado](#)

[Reglas](#)

Puntos de conexión globales

Archivos

Reproducciones

Expresión Cron

0 2 * * *

Fechas de los siguientes 10 desencadenadores

Zona horari...

jue, 16 oct 2025, 21:00 GMT-5
vie, 17 oct 2025, 21:00 GMT-5
sáb, 18 oct 2025, 21:00 GMT-5
dom, 19 oct 2025, 21:00 GMT-5
lun, 20 oct 2025, 21:00 GMT-5
mar, 21 oct 2025, 21:00 GMT-5
mié, 22 oct 2025, 21:00 GMT-5
jue, 23 oct 2025, 21:00 GMT-5
vie, 24 oct 2025, 21:00 GMT-5
sáb, 25 oct 2025, 21:00 GMT-5

Paso 3: seleccionar los destinos [Editar](#)

Como resultado se hace la ingesta automatiza en s3 y se ejecuta todo el proceso hasta la publicación en la vista y consumo en el informe de manera automática:

Objetos (21)

[Copiar URI de S3](#) [Copiar URL](#) [Descargar](#) [Abrir](#) [Eliminar](#) [Acciones](#)

[Crear carpeta](#) [Cargar](#)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	orders_2012-12.csv	csv	16 Oct 2025 8:50:34 PM -05	200.9 KB	Estándar
<input type="checkbox"/>	orders_2013-01.csv	csv	16 Oct 2025 9:00:58 PM -05	143.4 KB	Estándar

© 2025, Amazon Web Services, Inc. o sus filiales. [Privacidad](#) [Términos](#) [Preferencias de cookies](#)

21:04 16/10/2025

aws [Buscar](#) [Alt+S] Estados Unidos (Norte de Virginia) ID de cuenta: 2557-3143-1472 BG2003

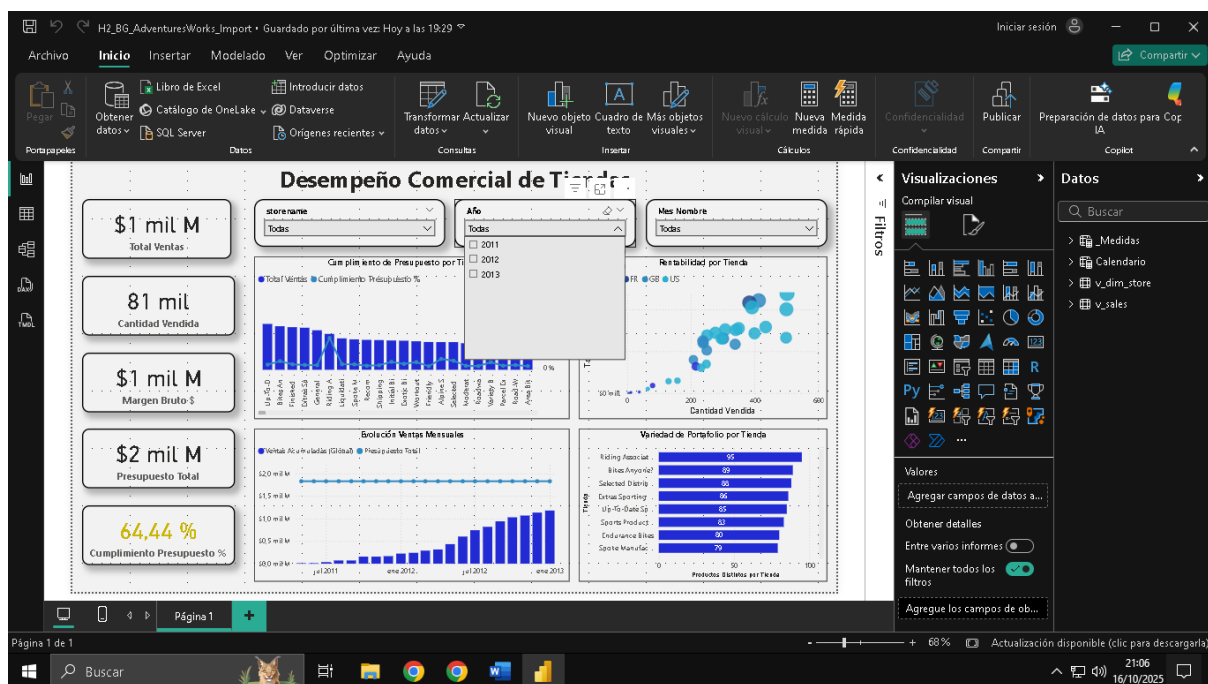
[Amazon Athena](#) > Editor de consultas

#	RunMonth	ventas
10	2012-05	1.1847989874750015E8
11	2012-06	1.6819547926989946E8
12	2012-07	1.4083522928060016E8
13	2012-08	7.517371221640076E7
14	2012-09	1.2186316565909943E8
15	2012-10	8.862330735659955E7
16	2012-11	3.91627642049997E7
17	2012-12	6.549763290270053E7
18	2013-01	4.513503106970003E7

CloudShell Comentarios

© 2025, Amazon Web Services, Inc. o sus filiales. [Privacidad](#) [Términos](#) [Preferencias de cookies](#)

21:05 16/10/2025



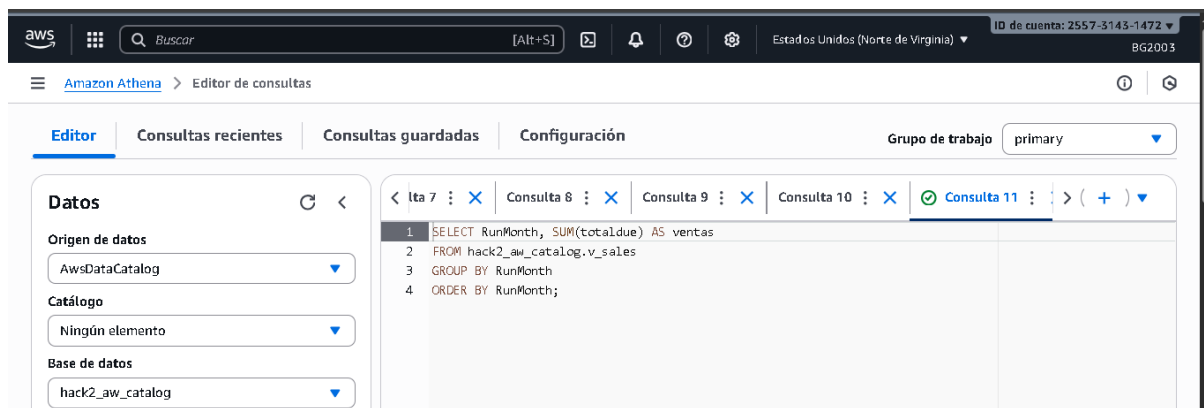
3.7. Creación de vistas analíticas y conexión con Power BI

Con las tablas finales creadas y el proceso incremental funcionando, se construyeron vistas en Amazon Athena para facilitar el consumo analítico desde Power BI y simplificar las relaciones del modelo de datos. Las vistas creadas fueron las siguientes:

- **v_sales:** representó una Big Table que combina la tabla de hechos fact_sales con las dimensiones principales (clientes, empleados, productos y tiendas), consolidando todos los indicadores comerciales en una sola estructura analítica, manteniendo OrderDate como referencia temporal.
- **v_dim_store:** expone las tiendas junto con su presupuesto total de ventas. El campo *Budget* representa la meta acumulada de ventas para todo el período (2011–2014), permitiendo calcular indicadores de cumplimiento en Power BI.
- **v_calendar_month_loaded:** genera un calendario dinámico basado exclusivamente en los meses cargados en la tabla fact_sales. Esto garantiza que el modelo solo muestre los períodos que existen en el Data Lake, evitando fechas vacías o sin ventas.

Estas vistas se registraron dentro de la base de datos **hack2_aw_catalog**, almacenadas también en formato Parquet dentro de la carpeta *warehouse*, y fueron utilizadas como base de conexión en Power BI.

El código usado se encuentra disponible en: https://github.com/BryanGuapulema/Hackaton-2/tree/main/vistas_powerBI



Datos

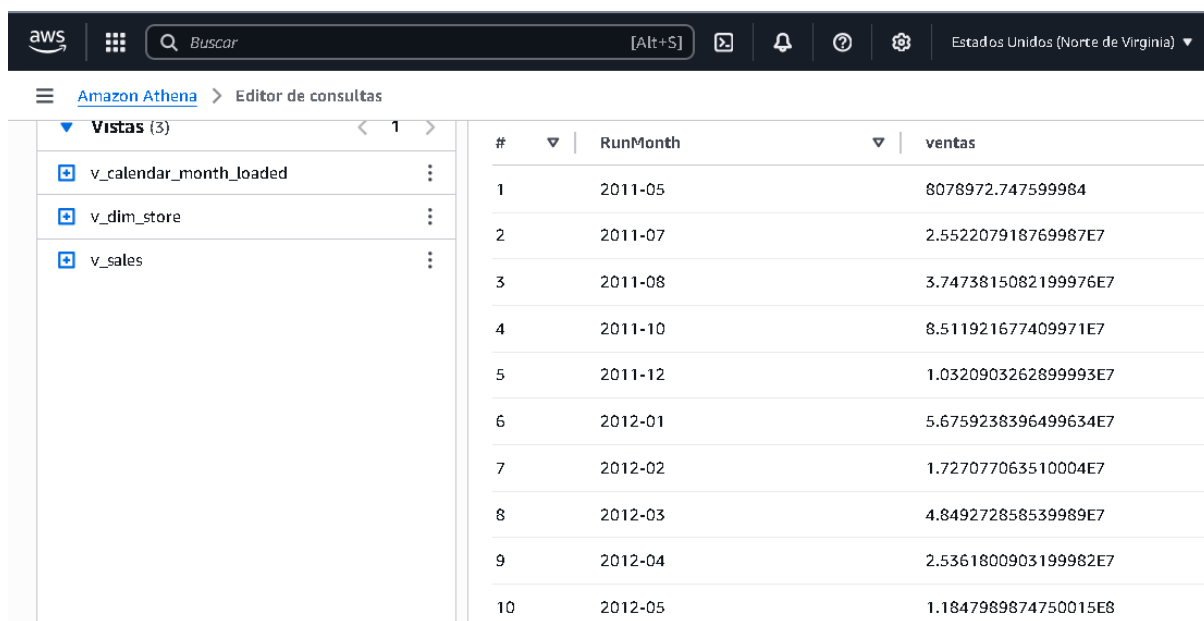
Origen de datos: AwsDataCatalog

Catálogo: Ningún elemento

Base de datos: hack2_aw_catalog

Consulta 11:

```
1 SELECT RunMonth, SUM(totaldue) AS ventas
2 FROM hack2_aw_catalog.v_sales
3 GROUP BY RunMonth
4 ORDER BY RunMonth;
```



#	RunMonth	ventas
1	2011-05	8078972.747599984
2	2011-07	2.552207918769987E7
3	2011-08	3.7473815082199976E7
4	2011-10	8.511921677409971E7
5	2011-12	1.0320903262899993E7
6	2012-01	5.6759238396499634E7
7	2012-02	1.727077063510004E7
8	2012-03	4.849272858539989E7
9	2012-04	2.5361800903199982E7
10	2012-05	1.1847989874750015E8

4. Visualización

4.1. Conexión a Power BI

Para la visualización, se configuró el conector ODBC de Amazon Athena, siguiendo estos pasos:

- **Creación del usuario IAM:** Se creó un usuario con permisos sobre S3, Glue y Athena, asignándole políticas de acceso:
 - S3-access-PowerBI
 - Glue-access-PowerBI
 - Athena-access-PowerBI

Nota: Se generó una Access Key y Secret Key para autenticación directa.

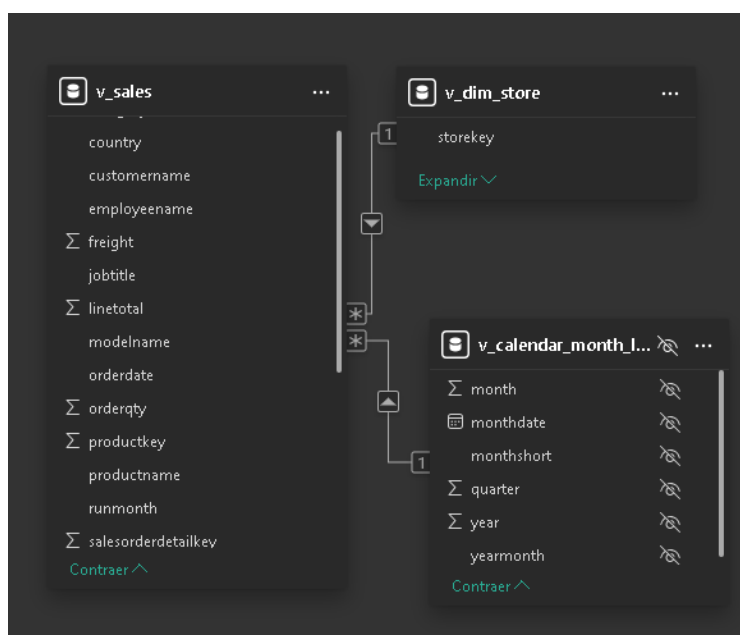
- **Configuración del DSN en ODBC Administrator:** Se configuró la conexión indicando:
 - Región: us-east-1
 - Catalog: AwsDataCatalog

- Database: hack2_aw_catalog
- Authentication: Access Key & Secret Key.
- **Conexión desde Power BI:** En Power BI Desktop se buscó la opción de conexión a Amazon Athena (ODBC) y se usó el DSN creado.

Se eligió el modo Import, que permite mayor rendimiento y refrescos del modelo cuando se actualiza con los nuevos meses cargados por la lambda incremental.

4.2. Modelo de datos

Tras conectar con las vistas en Power BI Desktop, se construyó a partir de las tres vistas (v_sales, v_dim_store y v_calendar_month_loaded) y se definieron las relaciones entre tablas, resultando en el esquema presentado enseguida:



4.3. Medidas DAX

Se definió la siguiente lista de medidas dentro de una tabla contenedora de medidas:

Medida	Fórmula
% Participación en Ventas	% Participación en Ventas = DIVIDE([Total Ventas], CALCULATE([Total Ventas], ALL(v_sales)))
Cantidad Vendida	Cantidad Vendida = SUM(v_sales[orderqty])
Costo	Costo = SUM(v_sales[freight])+SUM(v_sales[taxamt])
Cumplimiento Presupuesto %	Cumplimiento Presupuesto % = DIVIDE([Total Ventas], [Presupuesto Total],0)
Margen Bruto \$	Margen Bruto \$ = [Total Ventas] - [Costo]
Presupuesto Total	Presupuesto Total = SUM(v_dim_store[budgetamt])
Productos Distintos por Tienda	Productos Distintos por Tienda = DISTINCTCOUNT(v_sales[productkey])
Total Ventas	Total Ventas = SUM(v_sales[totaldue])
Ventas Acumuladas (Global)	Ventas Acumuladas (Global) = VAR MaxMes = MAX (Calendario[MonthDate])

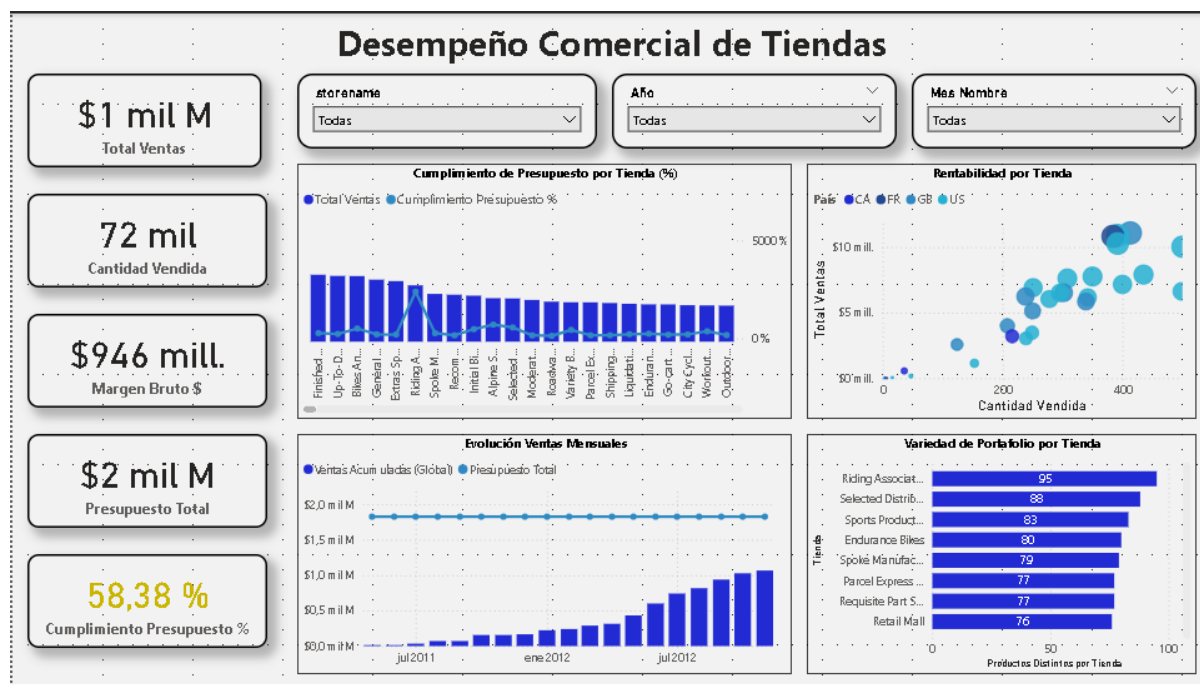

```

RETURN
CALCULATE (
    [Total Ventas],
    FILTER (
        ALL ( Calendario[MonthDate] ),
        Calendario[MonthDate] <= MaxMes
    )
)

```

5. Visualización

5.1. Dashboard



5.2. Historia construida a partir del dashboard

Este proyecto analiza el rendimiento comercial de Adventure Works, una empresa distribuidora de productos y accesorios de ciclismo, durante el periodo comprendido entre mayo de 2011 y mayo de 2014. El objetivo principal fue medir el cumplimiento de las metas de venta de cada tienda frente a su presupuesto total, así como analizar la rentabilidad y diversidad de portafolio en los diferentes puntos de venta.

Comenzando por la parte superior izquierda, observamos las principales métricas globales. Durante los tres años analizados, Adventure Works alcanzó ventas totales de aproximadamente \$1.000 millones, con 72 mil unidades vendidas y un margen bruto de \$946 millones, lo que evidencia una estructura de costos eficiente y una alta rentabilidad. El presupuesto total asignado a todas las tiendas fue de cerca de \$2.000 millones, de los cuales se alcanzó un cumplimiento promedio del 58,4%. Este indicador revela que, aunque la empresa mantiene una fuerte base de ventas, aún existe un amplio margen para optimizar la ejecución presupuestaria y la asignación de metas por tienda.

En el gráfico superior central se observa el porcentaje de cumplimiento del presupuesto por cada tienda. Se nota una fuerte disparidad entre las tiendas: algunas sobrecumplen sus metas por amplios márgenes, mientras otras no logran superar el 10% de su presupuesto. Esto demuestra una distribución ineficiente de objetivos, donde algunas tiendas tienen presupuestos poco realistas frente a su potencial real de venta. Una recomendación clara sería recalibrar las metas comerciales en función del histórico de ventas y la variedad de productos que cada tienda maneja.

En el gráfico superior derecho, se visualiza la relación entre la cantidad vendida y el valor total de ventas, segmentada por país. Las burbujas de mayor tamaño corresponden a las tiendas más rentables. Se observa que la mayor rentabilidad se concentra en las tiendas de Estados Unidos (US), que además presentan un alto volumen de ventas. Este patrón sugiere que concentrar las estrategias comerciales en este mercado podría maximizar la eficiencia del presupuesto y mejorar el retorno global. En contraste, las tiendas de menor volumen, especialmente en otras regiones, presentan márgenes reducidos y menor impacto en la rentabilidad general.

En la parte inferior central, el gráfico muestra la evolución de las ventas acumuladas a lo largo del tiempo frente al presupuesto total. Se evidencia un crecimiento sostenido, especialmente entre 2011 y 2012, lo que indica una expansión saludable del negocio. Sin embargo, en algunos tramos, las ventas se estabilizan sin alcanzar el ritmo del presupuesto, reflejando períodos en los que la ejecución comercial fue más lenta. Este análisis temporal permite identificar meses de alto desempeño y meses de baja actividad, información crucial para planificar estrategias promocionales o campañas de impulso.

Finalmente, en el gráfico inferior derecho se presenta la variedad de productos vendidos por tienda. Se observa que las tiendas con un portafolio más amplio tienden también a tener mayores volúmenes de venta y mejor cumplimiento presupuestario. Esto demuestra que la diversificación de productos es un factor clave de éxito comercial: las tiendas que ofrecen más variedad tienden a fidelizar mejor a los clientes y mantener ventas sostenidas.

En conjunto, el dashboard refleja que Adventure Works es una empresa rentable, con una fuerte concentración de ventas en Estados Unidos y un portafolio amplio, pero con una planificación presupuestaria desigual entre tiendas. Las oportunidades de mejora se centran en tres líneas: reestructurar los presupuestos por tienda, basados en su potencial de ventas y su portafolio real, consolidar las operaciones en los mercados más rentables (principalmente Estados Unidos), impulsar la variedad de productos en tiendas con bajo desempeño, ya que esto se traduce directamente en mayor volumen de ventas.

6. Mejores prácticas utilizadas

Durante el desarrollo del proyecto se identificaron y aplicaron diversas buenas prácticas técnicas recomendadas para la plataforma AWS, las cuales optimizaron el rendimiento, la trazabilidad y los costos del entorno.

a) Diseño del Data Lake y seguridad

- Se bloqueó el acceso público al bucket S3 y se habilitó el cifrado automático (SSE-S3) para proteger los datos almacenados.

- Se organizó la estructura del bucket de forma modular, separando bronze, warehouse y logs para facilitar el mantenimiento y auditoría.
- Se evitó la sobrescritura de archivos, manteniendo histórico por mes de carga (ej. orders_YYYY-MM.csv).

b) Uso de servicios sin servidor (Serverless)

- El proyecto utilizó servicios serverless (Lambda, Athena, DynamoDB, EventBridge), eliminando la necesidad de administrar infraestructura y reduciendo costos.
- Las funciones Lambda se diseñaron con time-out controlado (5 minutos) y registro centralizado en CloudWatch, garantizando ejecuciones seguras y trazables.
- Se empleó DynamoDB en modo On-Demand, evitando configuración manual de capacidad y asegurando escalabilidad automática.

c) Control de logs y auditoría

- Se centralizaron los logs operativos y de errores en DynamoDB, permitiendo monitoreo detallado por run_id, tabla y estado.
- Los resultados de Athena se almacenaron en S3/logs/athena-results/, asegurando persistencia y trazabilidad de cada consulta SQL ejecutada.
- CloudWatch se utilizó para la depuración y seguimiento de errores de ejecución en las Lambdas.

d) Automatización modular

- Se estableció un trigger S3 para invocar automáticamente la Lambda h2-factsales-upsert-month al detectar nuevos archivos de órdenes mensuales. Esto permitió la actualización incremental sin intervención manual.
- Se planificó la integración de Amazon EventBridge para la ejecución diaria automatizada de la Lambda ingest_csv_github, programada para generar un nuevo mes cada día. De esta forma, el sistema avanza automáticamente por meses (ej. de 2012-11 → 2012-12 → 2013-01, etc.), asegurando un proceso continuo y sostenible.

e) Optimización y consumo analítico

- Se empleó el formato Parquet para las tablas del warehouse, reduciendo el tamaño de almacenamiento y mejorando el rendimiento de las consultas en Athena.
- Se utilizaron vistas analíticas para abstraer la complejidad del modelo y permitir un consumo directo desde Power BI sin afectar el Data Lake.
- En Power BI se optó por el modo Import, con actualizaciones manuales tras nuevas cargas, optimizando la velocidad de visualización y análisis.

f) Escalabilidad y costo-eficiencia

- Todos los servicios utilizados pertenecen al Free Tier de AWS, aprovechando al máximo los recursos gratuitos disponibles.

- Se evitó el uso de servicios de alto costo como Glue Crawlers o instancias EC2, priorizando soluciones ligeras y nativas sin servidor.
- La arquitectura es fácilmente escalable a medida que se agregan más meses o años de datos sin modificar la estructura base.

7. Recursos

- **Documentación completa:** <https://github.com/BryanGuapulema/Hackaton-2>
- **Fuentes de datos CSV y Excel:** https://github.com/BryanGuapulema/AW_data_csv