1. Introducción

El presente documento tiene como objetivo describir de forma detallada el desarrollo de un proyecto de procesamiento ETL (Extracción, Transformación y Carga) y análisis OLAP (Procesamiento Analítico en Línea) aplicado a la base de datos AdventureWorks, una base de datos de ejemplo proporcionada por Microsoft que simula un entorno empresarial real orientado al comercio de productos.

Este proyecto surge como una iniciativa para poner en práctica las etapas fundamentales de la inteligencia de negocios, integrando el uso de SQL Server para la manipulación y almacenamiento de datos, Visual Studio con herramientas de Integration Services (SSIS) y Analysis Services (SSAS) para la automatización de flujos de datos y la creación de cubos OLAP, y Power BI para la visualización e interpretación de los resultados.

El proyecto abarca desde la extracción de los datos desde AdventureWorks, su transformación en una base intermedia, la carga de datos transformados a un almacén de datos (Data Warehouse) y la posterior construcción de un cubo OLAP para realizar análisis multidimensionales sobre ventas, productos, tiempo, regiones y otros factores relevantes.

Este proceso representa un ciclo completo de implementación de un sistema de inteligencia de negocios que puede ser aplicado en contextos empresariales reales para mejorar la toma de decisiones basada en datos.

2. Objetivos

Objetivo General

Desarrollar un proceso completo de integración y análisis de datos mediante una arquitectura ETL, utilizando como fuente la base de datos AdventureWorks, para

construir un Data Warehouse optimizado que permita realizar análisis OLAP a través de un cubo multidimensional y visualizaciones interactivas en Power BI.

Objetivos específicos

- Diseñar una base de datos intermedia (ventasStage) que permita la estandarización y transformación de datos provenientes de AdventureWorks.
- Implementar procesos de extracción, transformación y carga (ETL) utilizando SQL Server Integration Services (SSIS).
- Construir un Data Warehouse (ventasDWH) con un modelo en estrella que permita el análisis de ventas y otras métricas clave.
- Crear un cubo OLAP mediante SQL Server Analysis Services (SSAS) que habilite el análisis multidimensional.
- Desarrollar dashboards en Power BI conectados al cubo OLAP para facilitar la toma de decisiones basada en datos.

3. Herramientas utilizadas

A lo largo del desarrollo del proyecto se emplearon diversas herramientas tecnológicas para cubrir cada fase del proceso ETL y del análisis OLAP. A continuación, se describen:

Microsoft SQL Server 2019

Utilizado para gestionar las bases de datos relacionalmente.

- Alojó las tres instancias principales del proyecto:
- Base de datos fuente: AdventureWorks2019
- Base de datos intermedia: ventasStage
- Data Warehouse: ventasDWH

SQL Server Integration Services (SSIS)

Usado mediante Visual Studio para diseñar los flujos de extracción, transformación y carga.

Se desarrollaron tres paquetes:

- Extracción desde AdventureWorks hacia ventasStage.
- Transformación para limpiar y estructurar los datos en ventasStage.
- Carga de datos transformados hacia ventasDWH.

SQL Server Analysis Services (SSAS)

- Herramienta de análisis multidimensional utilizada para construir el cubo OLAP.
- Permite crear dimensiones, medidas y particiones sobre los datos del Data Warehouse.
- Se diseñó un cubo con base en la tabla de hechos FACT_Ventas.

Power BI

- Plataforma de visualización de datos conectada al cubo OLAP.
- Se diseñaron dashboards interactivos que permiten explorar las ventas desde múltiples perspectivas (producto, tiempo, cliente, etc.).

4. Diseño de sistemas ETL

El proceso ETL (Extracción, Transformación y Carga) se diseñó para extraer datos relevantes de la base AdventureWorks2019, transformarlos adecuadamente en una base staging intermedia (ventasStage) y finalmente cargarlos en un Data Warehouse optimizado (ventasDWH). Este diseño asegura la limpieza, integridad y trazabilidad de los datos a lo largo del proceso.

4.1 Base de datos fuente: AdventureWorks

AdventureWorks es una base de datos de ejemplo proporcionada por Microsoft que simula el entorno de una empresa manufacturera y de ventas. Las tablas clave utilizadas incluyen:

- Sales.Customer y Person.Person para clientes.
- Sales.SalesPerson para vendedores.
- Sales.SalesOrderHeader y Sales.SalesOrderDetail para órdenes y detalles de ventas.
- Production.Product, ProductModel, ProductSubcategory, ProductCategory para catálogo de productos.

Para facilitar la extracción, se crearon vistas SQL como:

- vclientes, vvendedor, vfechas – que unifican y transforman datos básicos antes de insertarlos en la base intermedia.

4.2 Base de datos intermedia: Ventas Stage

La base intermedia ventasStage contiene tablas STG (staging) que conservan los datos transformados desde AdventureWorks. Su diseño incluye:

- Generación de claves sustitutas (SK) mediante columnas IDENTITY para todas las tablas.
- Relaciones entre entidades: producto → subcategoría → categoría, vendedor → orden, etc.
- Tablas clave: STG_producto, STG_orden, STG_detalleOrden, STG_cliente, STG_modelo, STG_subcategoria, STG_categoria, STG_fechas, STG_vendedor, STG_territorio.

Procedimientos aplicados en Ventas Stage

1.- Limpia tablas: Elimina todos los registros de las tablas STG y reinicia los contadores IDENTITY mediante DBCC CHECKIDENT.

```
delete from STG_DetalleOrden;
delete from STG_producto;
delete from STG_modelo;
delete from STG_subcategoria;
delete from STG_categoria;
delete from STG_Orden;
delete from STG_fechas;
delete from STG_territorio;
delete from STG_cliente;
delete from STG_vendedor;

DBCC checkident(stg_categoria,reseed,0);
-- (continúa para las demás tablas...)
```

2.- Tansforma categoría, tansforma subcategoria y transforma modelo

```
Insertan registros “por defecto” con ID = 0 en caso de datos nulos o faltantes.
if not exists (select * from STG_categoria where categoriaID = 0)
    insert into STG_categoria values(0,'sin categoria');
if not exists (select * from STG_subcategoria where subcategoriaID = 0)
    insert into STG_subcategoria values(0,'sin categoria',0);
if not exists (select * from STG_modelo where modeloID = 0)
    insert into STG_modelo values(0,'sin modelo');
```

3.- Transformar producto

Rellena valores nulos de color, modelo y subcategoría y calcula y actualiza los SK correspondientes mediante subconsultas.

```
update STG_producto set color = 'Sin color' where color is null;

update STG_producto set subcategoriaID = 0 where subcategoriaID is null;
```

```
update STG_producto set modeloID = 0 where modeloID is null;


update STG_producto
set SKsubcategoriaID = (select SKsubcategoriaID from STG_subcategoria
                        where              STG_subcategoria.subcategoriaID              =
STG_producto.subcategoriaID)
where SKsubcategoriaID is null;


-- (continúa para SKmodeloID y SKcategoriaID)
```

4.- transformaVendedor, transformaOrden, transformaDetalleOrden

Asignan valores por defecto a vendedores faltantes (“ventas en línea”).

Calculan los SK de cliente, vendedor, fecha, territorio y producto para las órdenes y sus detalles.

```
update STG_Orden set vendedorID = 0 where vendedorID is null;
update STG_Orden
set SKclienteID = (select SKclienteID from STG_cliente where STG_Orden.clienteID =
STG_cliente.clienteID)
where SKclienteID is null;
-- (continúa para SKvendedorID, SKterritorioID, SKfecha)
update STG_DetalleOrden
set SKproductoid = (select SKproductoid from STG_producto
                    where STG_DetalleOrden.productoid = STG_producto.productoID)
where SKproductoid is null;
```

4.3 Base de datos Data Warehouse: ventasDWH

ventasDWH representa el almacenamiento central del proyecto con un modelo en estrella. Contiene:

- Dimensiones: DIM_producto, DIM_cliente, DIM_fecha, DIM_vendedor, DIM_territorio, DIM_categoria, DIM_subcategoria, DIM_modelo.

- Tabla de hechos: FACT_Ventas que almacena las métricas de negocio.

La tabla de hechos calcula:

- costoUnitario, costoVentas, utilidad, descuento, totalVentas.

Procedimientos aplicados en ventasDWH:

1.- limpiaTablas

- Elimina registros en todas las dimensiones y hechos para facilitar recargas limpias.

```
delete from FACT_Ventas;
delete from DIM_fecha;
delete from DIM_producto;
delete from DIM_categoria;
delete from DIM_subcategoria;
delete from DIM_modelo;
delete from DIM_cliente;
delete from DIM_vendedor;
delete from DIM_territorio;
```

2.- vw_FactVentasStage (vista)

- Une las tablas STG necesarias y calcula las métricas derivadas.
- Aplica divisiones seguras con NULLIF para evitar errores por división entre cero.

```
SELECT
    o.ordenID AS SKVenta,
    d.OrdenDetalleID AS detalleId,
    d.cantidad,
    (d.totalVentas / NULLIF(d.cantidad, 0)) AS costoUnitario,
    0 AS descuento,
    d.precioUnitario,
    d.totalVentas,
```

```
    d.cantidad * (d.totalVentas / NULLIF(d.cantidad, 0)) AS costoVentas,
    d.totalVentas - (d.cantidad * (d.totalVentas / NULLIF(d.cantidad, 0))) AS utilidad,
    f.SKfecha,
    p.SKproductoID,
    ca.SKcategoriaID,
    sc.SKsubcategoriaID,
    m.SKmodeloID,
    c.SKclienteID,
    ve.SKvendedorID,
    t.SKterritorioID
FROM STG_Orden AS o
-- (joins con STG_* omitidos por brevedad)
```

3.- sp_InsertFactVentas

- Inserta los datos desde la vista en FACT_Ventas, incluyendo claves sustitutas y valores calculados.

```
INSERT INTO FACT_Ventas (
    ordenID,
    detalleId,
    cantidad,
    costoUnitario,
    descuento,
    precioUnitario,
    totalVentas,
    costoVentas,
    utilidad,
    SKfecha,
    SKproductoID,
    SKcategoriaID,
    SKsubcategoriaID,
    SKmodeloID,
```

```
    SKclienteID,
    SKvendedorID,
    SKterritorioID
)
SELECT
    SKVenta,
    detalleId,
    cantidad,
    costoUnitario,
    0,
    precioUnitario,
    totalVentas,
    costoVentas,
    utilidad,
    SKFecha,
    SKProductoID,
    SKCategoriaID,
    SKSubcategoriaID,
    SKModeloID,
    SKClienteID,
    SKVendedorID,
    SKTerritorioID
FROM vw_FactVentasStage;
```

5. Procesos SSIS

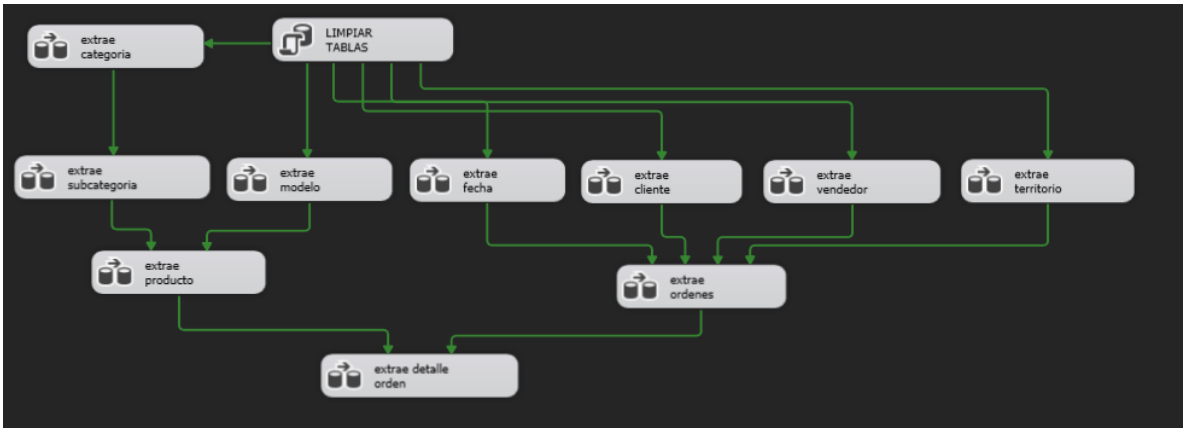
Para implementar el flujo ETL de manera visual y automatizada, se desarrollaron tres paquetes SSIS utilizando Visual Studio. Cada uno corresponde a una fase del proceso: extracción, transformación y carga. A continuación, se describe cada uno de ellos.

5.1 Extraccion

Objetivo: Extraer datos limpios y organizados desde la base de datos AdventureWorks2019 hacia la base intermedia ventasStage.

Elementos clave:

- **Origen de datos:** tablas y vistas de AdventureWorks2019.
- **Destino:** tablas de ventasStage.
- **Componentes usados:**
    - Tareas Data Flow Task para cada entidad: cliente, vendedor, fecha, producto, modelo, subcategoría, categoría, orden, detalle de orden.
    - Conversión de tipos de datos si es necesario.
    - Inserción directa en las tablas STG usando OLE DB Destination.



5.2 Transformación

Objetivo: Limpiar, completar y transformar los datos en ventasStage antes de su carga al Data Warehouse.

Componentes usados:

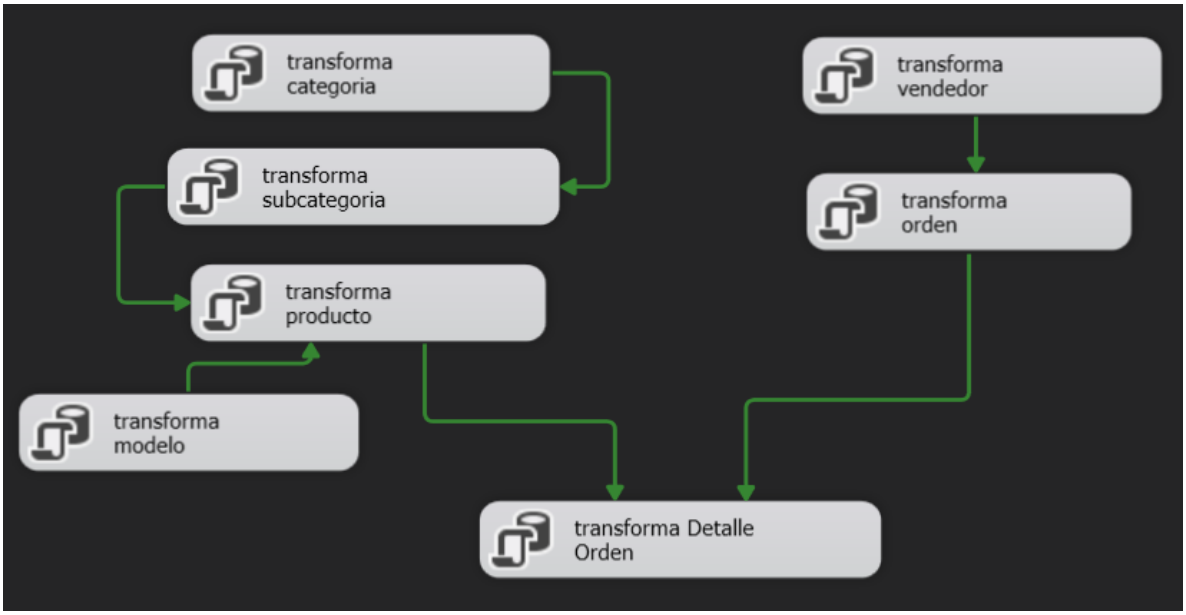
- Tarea Execute SQL Task para ejecutar procedimientos como:
    - transformaCategoria
    - transformaSubCat
    - transformaModelo
    - transformaProducto

- transformaVendedor
- transformaOrden
- transformaDetalleOrden

- Tarea Sequence Container para organizar los pasos.

Ventajas:

- Los datos incompletos se completan con valores por defecto (0, Sin color, etc.).
- Las claves sustitutas (SK) se actualizan internamente asegurando consistencia relacional.



5.3 Carga

Objetivo: Cargar los datos transformados desde ventasStage hacia el Data Warehouse ventasDWH.

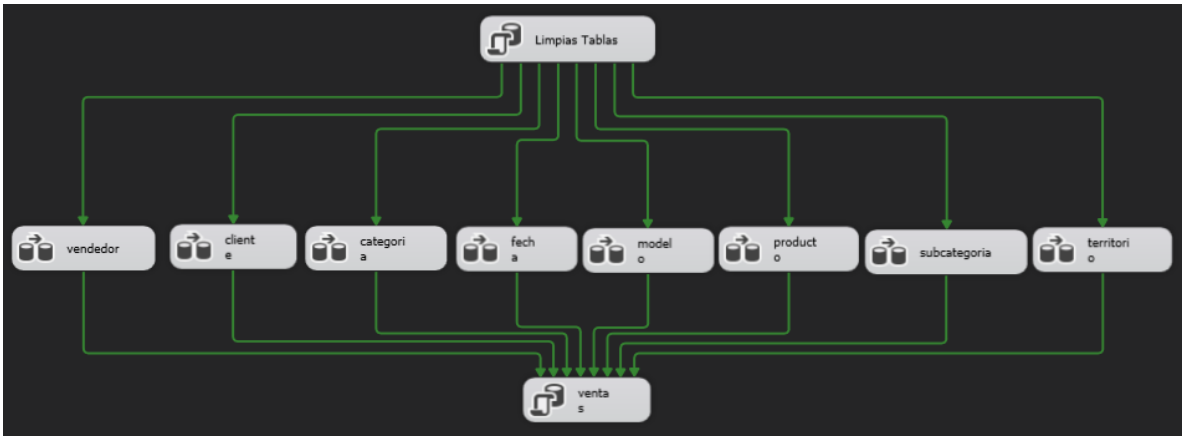
Componentes usados:

- Tarea Execute SQL Task para ejecutar:
  - limpiaTablas (elimina datos previos del DWH).

    - sp_InsertFactVentas (inserta registros en FACT_Ventas).
- Transferencia de datos directa desde la vista vw_FactVentasStage.

Ventajas:

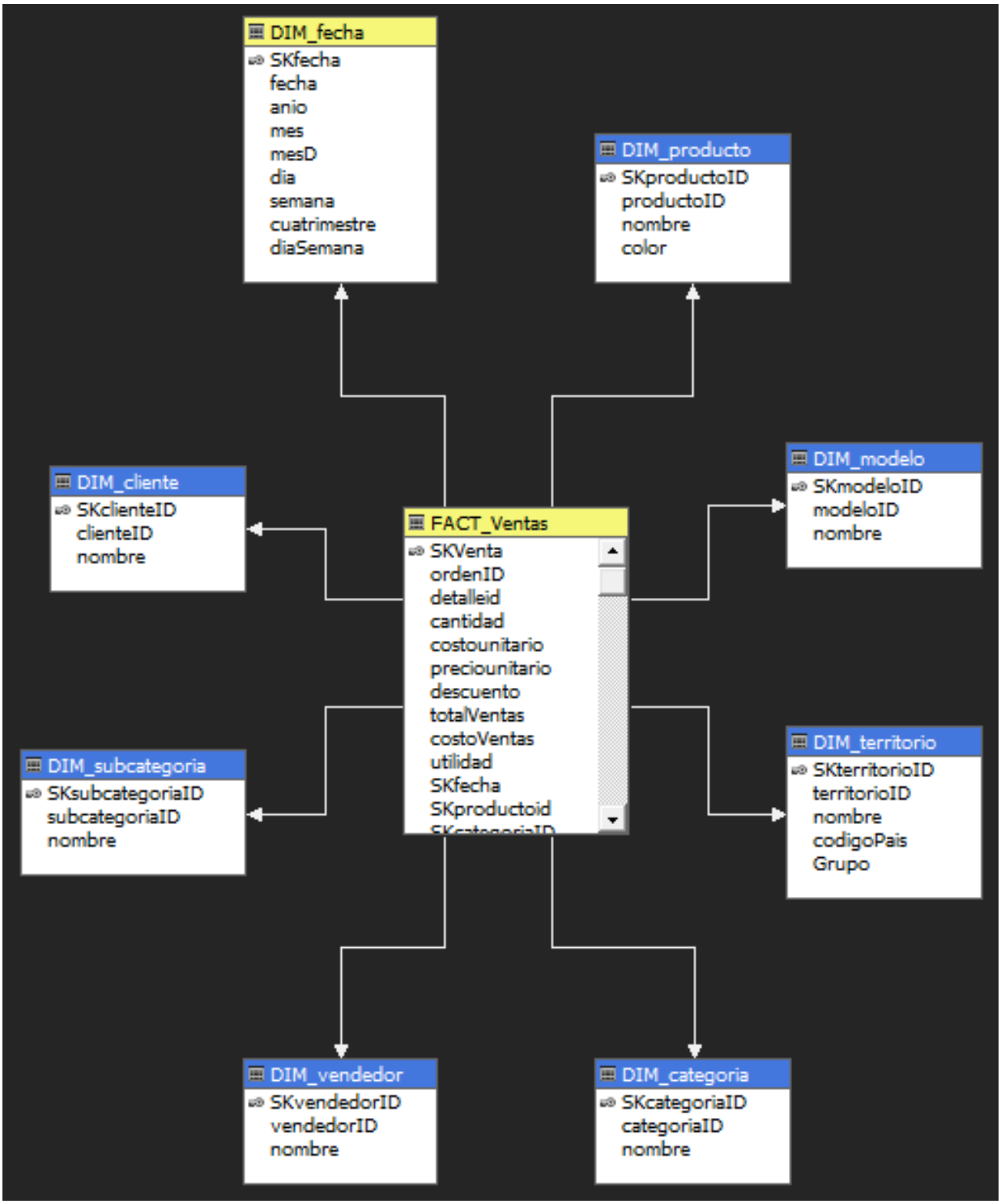
- Se mantiene limpia la tabla de hechos con cada ejecución.
- Permite repetir cargas sin duplicar datos.



6. Construcción del cubo OLAP

El análisis multidimensional del Data Warehouse se implementó mediante un cubo OLAP utilizando SQL Server Analysis Services (SSAS), aprovechando la base ventasDWH como origen de datos. Esta solución permite explorar las métricas de ventas desde múltiples perspectivas, como tiempo, producto, cliente o territorio.

6.1 Proyecto de anlysis services


DIM_fecha
SKfecha
fecha
anio
mes
mesD
dia
semana
cuatrimestre
diaSemana
DIM_producto
SKproductoID
productoID
nombre
color
DIM_cliente
SKclienteID
clienteID
nombre
FACT_Ventas
SKVenta
ordenID
detalleid
cantidad
costounitario
preciounitario
descuento
totalVentas
costoVentas
utilidad
SKfecha
SKproductoid
DIM_modelo
SKmodeloID
modeloID
nombre
DIM_subcategoria
SKsubcategoriaID
subcategoriaID
nombre
DIM_territorio
SKterritorioID
territorioID
nombre
codigoPais
Grupo
DIM_vendedor
SKvendedorID
vendedorID
nombre
DIM_categoria
SKcategoriaID
categoriaID
nombre

6.2. Dimensiones creadas

Dimensión	*Archivo*	*Atributos principales*
Producto	DIM Producto.dim	nombre, color
Categoría	DIM Categoria.dim	nombre
Subcategoría	DIM Subcategoria.dim	nombre
Modelo	DIM Modelo.dim	nombre
Cliente	DIM Cliente.dim	nombre
Vendedor	DIM Vendedor.dim	nombre
Territorio	DIM Territorio.dim	grupo, código de país
Fecha	DIM Fecha.dim	año, mes, día, cuatrimestre, día de la semana

7. Visualizacion en PowerBI

El siguiente ejemplo es solamente una pequeña demostración de lo que se puede lograr con la capacidad narrativa de PowerBI con un buen analisis de datos logrado a través del proceso ETL.

El dashboard es completamente interactivo, lo que significa que si quiero seleccionar un año especifico, con un pais especifico puedo ver las ventas totales de ese año, en ese pais y las posibilidades van mas allá de esto:

Incluso con los datos históricos que tenemos podemos tener la capacidad de detectar trends en la venta dependiendo del mes e incluso si un cliente tiende a comprar en una cierta época del año se puede predecir con herramientas estadísticas.

8. Conclusión

A través de la creación de una base de datos intermedia, se logró normalizar, transformar y limpiar datos provenientes de múltiples tablas relacionales, asegurando su integridad y trazabilidad. La implementación de claves sustitutas y el uso de procedimientos almacenados permitieron enriquecer los datos y prepararlos adecuadamente para su análisis posterior.

Posteriormente, los datos fueron cargados a una base estructurada bajo un modelo dimensional en estrella , permitiendo segmentar la información en dimensiones claves como producto, tiempo, cliente, vendedor y territorio. La tabla de hechos FACT_Ventas consolidó las métricas más relevantes para el negocio, como la cantidad de ventas, costos, utilidad y descuentos.

Con base en esta arquitectura, se construyó un cubo OLAP mediante SQL Server Analysis Services (SSAS), que habilitó el análisis multidimensional eficiente. Finalmente, se diseñó un dashboard en Power BI que facilitó la interpretación visual de los datos, permitiendo detectar patrones, identificar líderes de ventas, analizar la distribución geográfica y evaluar el rendimiento de productos y categorías.

Este proyecto demuestra el potencial de combinar técnicas de ingeniería de datos con herramientas de inteligencia de negocios, resultando en una solución sólida, escalable y orientada a la toma de decisiones basada en datos.