Evidencia de Aprendizaje 3

Proceso de Transformación de Datos

y Carga en el Data Mart Final

Justin Beckham Cardona

Jorge Armando Rodríguez

Lina Johana Seguro

Facultad de Ingeniería y Ciencias Agropecuarias,

Institución Universitaria Digital de Antioquia

064: Bases de Datos II

Antonio Jesús Valderrama

30 de septiembre de 2025

Introducción

En la actualidad, las organizaciones necesitan transformar grandes volúmenes de datos en información útil que facilite la toma de decisiones. Para lograrlo, se implementan procesos ETL (Extraer, Transformar y Cargar) que permiten extraer datos de las bases transaccionales, depurarlos, enriquecerlos y cargarlos en estructuras diseñadas para facilitar el análisis.

En este proyecto se crea un Data Mart a partir de la base de datos Jardinería usando un modelo estrella. Primero, los datos pasan por un entorno de Staging, donde se revisan y preparan para asegurar su calidad. Luego, la información organizada se carga en el Data Mart final, que está formado por tablas de dimensiones y una tabla de hechos. Esto permite hacer consultas y análisis sobre ventas, clientes, productos y periodos de tiempo.

Esta arquitectura mejora el rendimiento de las consultas, y también posibilita el cálculo de métricas clave, como la identificación del producto más vendido, la categoría con más productos y el año con mayores ventas. De esta forma, el Data Mart se convierte en una herramienta fundamental para analizar la información de la empresa y ayudar a tomar mejores decisiones.

Objetivos

Objetivo General:

Desarrollar un proceso ETL que migre los datos de la base Jardinería hacia un Data Mart en modelo estrella, asegurando información limpia, consistente y optimizada para el análisis de ventas, como la identificación del producto más vendido, las categorías con más productos y los años con mayores ventas.

Objetivos Específicos:

1. Verificar la disponibilidad y consistencia de la base de datos Staging previamente creada, asegurando que los datos estén listos para el proceso ETL.
2. Transformar y depurar los datos en Staging, aplicando validaciones y cálculos necesarios para asegurar su calidad y consistencia.
3. Cargar los datos transformados en el Data Mart con modelo estrella, garantizando la correcta integración de las dimensiones y la tabla de hechos.
4. Validar los resultados mediante consultas analíticas que permitan responder preguntas clave del negocio, como el producto más vendido, la categoría con más productos y los años con mayores ventas.

Preparación

Antes de iniciar el proceso de transformación y carga de datos hacia el Data Mart, fue necesario realizar una fase de preparación en dos aspectos fundamentales:

Revisión del Modelo Estrella

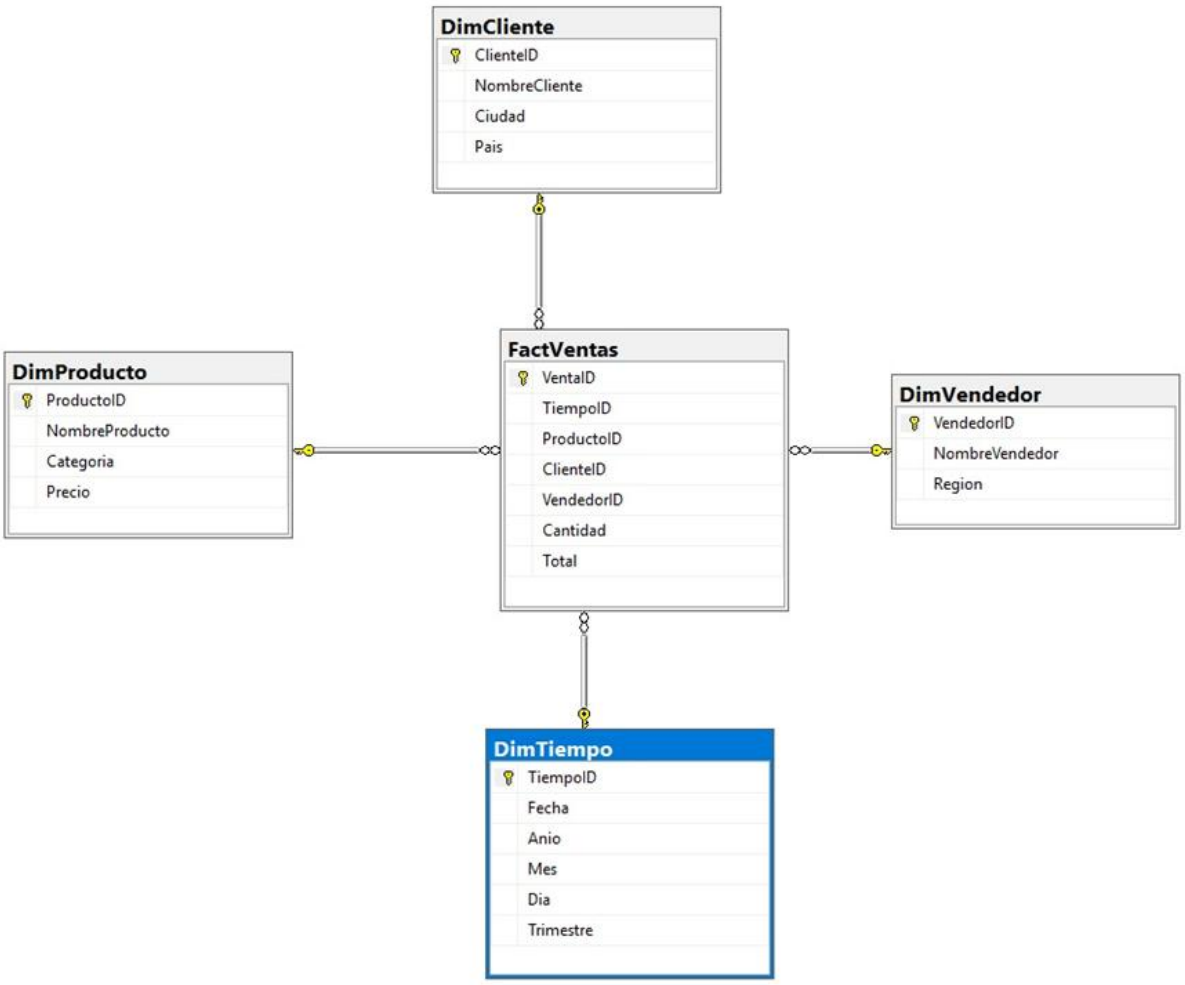
Se consultó el modelo estrella previamente definido en el proyecto de análisis, ya que permite estructurar los datos de forma clara para el análisis de ventas por cliente, producto, vendedor y tiempo.

En este caso donde trabajamos con la base de datos Jardinería, el modelo estrella definido se compone de las siguientes tablas de dimensiones (*DimCliente, DimProducto, DimVendedor, DimTiempo*) y la tabla de hechos ***(****FactVentas*).

- **DimCliente**: Contiene información de los clientes (ID, nombre, ciudad, país), lo que permite analizar las ventas por ubicación geográfica y cliente específico.
- **DimProducto**: Almacena datos de los productos (ID, nombre, categoría, precio), permitiendo clasificar y analizar las ventas según las características de los productos.
- **DimVendedor**: Incluye la información de los empleados encargados de las ventas (ID y nombre del vendedor, región), lo que posibilita analizar los resultados por vendedor o zona.
- **DimTiempo**: Descompone la fecha en atributos como año, mes, día y trimestre, para facilitar el análisis temporal de las ventas.

- **FactVentas**: Tabla de hechos que registra las transacciones de ventas, relacionando las dimensiones mencionadas mediante sus claves. Incluye métricas clave como la cantidad vendida y el valor total de las ventas.

El diseño del modelo estrella definido en el proyecto de análisis es el siguiente:



Esta revisión permitió comprender la estructura, las claves primarias y foráneas, así como las relaciones necesarias entre las dimensiones y los hechos, asegurando que los datos provenientes de la Staging pudieran integrarse correctamente en el Data Mart.

Verificación de la Disponibilidad de Base de Datos Staging

La base de datos **Jardineria_Staging** fue creada en la Actividad 2 y contiene las tablas necesarias para el proceso ETL. Se validó su existencia y disponibilidad mediante la consulta:

```
SELECT TABLE_SCHEMA AS Esquema,
       TABLE_NAME   AS Nombre_Tabla
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_TYPE = 'BASE TABLE'
ORDER BY TABLE_NAME;
```

	Esquema	Nombre_Tabla
1	dbo	stg_clientes
2	dbo	stg_detalle_pedido
3	dbo	stg_empleados
4	dbo	stg_oficinas
5	dbo	stg_pedidos
6	dbo	stg_productos

El resultado confirmó la presencia de las tablas maestras (*Stg_Cliente, Stg_Producto, Stg_Empleado, Stg_Oficina*) y las transaccionales (*Stg_Pedido, Stg_DetallePedido*).

Validaciones de Consistencia

Con el fin de asegurar la integridad de la información en la Staging, se realizaron las siguientes comprobaciones:

1. **Conteos origen vs. Staging**

   Se comparó la cantidad de registros entre la base Jardinería (origen) y la Staging:

Tabla 1
Conteo de registros: base origen vs. Staging

Tabla	Origen	Staging
Cliente	36	36
Producto	276	276
Pedido	115	115
Detalle_Pedido	318	318
Empleado	31	31
Oficina	9	9

Nota. Los valores fueron obtenidos mediante consultas COUNT(*) ejecutadas en SQL Server.

Los conteos de registros entre la base Jardinería (origen) y la base Jardineria_Staging coincidieron en todas las tablas, lo que confirma que la extracción e inserción de datos se realizaron de manera completa y sin pérdidas de información.

Nota: Consultas Sql de conteo de registros disponible en anexos.

2. Detección de huérfanos

Se validó que todos los pedidos tengan cliente y que todos los detalles de pedido tengan producto. No se encontraron registros huérfanos.

```sql
IF EXISTS (
    SELECT 1
    FROM stg_pedidos p
    LEFT JOIN stg_clientes c ON p.id_cliente = c.id_cliente
    WHERE c.id_cliente IS NULL
)
    SELECT 'Pedidos con cliente inexistente' AS Prueba,
           p.id_pedido, p.id_cliente
    FROM stg_pedidos p
    LEFT JOIN stg_clientes c ON p.id_cliente = c.id_cliente
    WHERE c.id_cliente IS NULL;
ELSE
    SELECT 'OK - Sin pedidos huérfanos' AS Mensaje;

IF EXISTS (
    SELECT 1
    FROM stg_detalle_pedido d
    LEFT JOIN stg_pedidos p ON d.id_pedido = p.id_pedido
    WHERE p.id_pedido IS NULL
)
    SELECT 'Detalle con pedido inexistente' AS Prueba,
           d.id_pedido, d.id_producto
    FROM stg_detalle_pedido d
    LEFT JOIN stg_pedidos p ON d.id_pedido = p.id_pedido
    WHERE p.id_pedido IS NULL;
ELSE
    SELECT 'OK - Sin detalles huérfanos (pedido)' AS Mensaje;
```

```sql
IF EXISTS (
    SELECT 1
    FROM stg_detalle_pedido d
    LEFT JOIN stg_productos p ON d.id_producto = p.id_producto
    WHERE p.id_producto IS NULL
)
    SELECT 'Detalle con producto inexistente' AS Prueba,
           d.id_pedido, d.id_producto
    FROM stg_detalle_pedido d
    LEFT JOIN stg_productos p ON d.id_producto = p.id_producto
    WHERE p.id_producto IS NULL;
ELSE
    SELECT 'OK - Sin detalles huérfanos (producto)' AS Mensaje;

IF EXISTS (
    SELECT 1
    FROM stg_empleados e
    LEFT JOIN stg_oficinas o ON e.id_oficina = o.id_oficina
    WHERE o.id_oficina IS NULL
)
    SELECT 'Empleados con oficina inexistente' AS Prueba,
           e.id_empleado, e.id_oficina
    FROM stg_empleados e
    LEFT JOIN stg_oficinas o ON e.id_oficina = o.id_oficina
    WHERE o.id_oficina IS NULL;
ELSE
    SELECT 'OK - Sin empleados huérfanos' AS Mensaje;
```

Resultados | Mensajes

	Mensaje
1	OK - Sin pedidos huérfanos

	Mensaje
1	OK - Sin detalles huérfanos (pedido)

	Mensaje
1	OK - Sin detalles huérfanos (producto)

	Mensaje
1	OK - Sin empleados huérfanos

3. Búsqueda de duplicados

Se verificó la unicidad de las claves primarias en cada tabla. No se encontraron duplicados en los identificadores principales.

```
IF EXISTS (
    SELECT 1
    FROM stg_clientes
    GROUP BY id_cliente
    HAVING COUNT(*) > 1
)
    SELECT 'Clientes duplicados' AS Prueba,
           id_cliente, COUNT(*) AS Total
    FROM stg_clientes
    GROUP BY id_cliente
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin clientes duplicados' AS Mensaje;
```

```
IF EXISTS (
    SELECT 1
    FROM stg_productos
    GROUP BY id_producto
    HAVING COUNT(*) > 1
)
    SELECT 'Productos duplicados' AS Prueba,
           id_producto, COUNT(*) AS Total
    FROM stg_productos
    GROUP BY id_producto
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin productos duplicados' AS Mensaje;
```

```
IF EXISTS (
    SELECT 1
    FROM stg_pedidos
    GROUP BY id_pedido
    HAVING COUNT(*) > 1
)
    SELECT 'Pedidos duplicados' AS Prueba,
           id_pedido, COUNT(*) AS Total
    FROM stg_pedidos
    GROUP BY id_pedido
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin pedidos duplicados' AS Mensaje;
```

```
IF EXISTS (
    SELECT 1
    FROM stg_detalle_pedido
    GROUP BY id_pedido, id_producto
    HAVING COUNT(*) > 1
)
    SELECT 'Detalle duplicado (pedido + producto)' AS Prueba,
           id_pedido, id_producto, COUNT(*) AS Total
    FROM stg_detalle_pedido
    GROUP BY id_pedido, id_producto
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin detalles duplicados' AS Mensaje;
```

Resultados | Mensajes

	Mensaje
1	OK - Sin clientes duplicados

	Mensaje
1	OK - Sin productos duplicados

	Mensaje
1	OK - Sin pedidos duplicados

	Prueba	id_pedido	id_producto	Total
1	Detalle duplicado (pedido + producto)	101	87	2

2. Chequeo de nulos en columnas críticas

Se revisó la presencia de valores nulos en columnas obligatorias como *id_cliente, id_producto, cantidad, precio_unidad y fecha_pedido*. Los resultados mostraron ausencia de nulos en estas columnas.

```
IF EXISTS (
    SELECT 1
    FROM stg_clientes
    WHERE nombre_cliente IS NULL OR ciudad IS NULL OR pais IS NULL
)
    SELECT 'Clientes con campos nulos' AS Prueba, *
    FROM stg_clientes
    WHERE nombre_cliente IS NULL OR ciudad IS NULL OR pais IS NULL;
ELSE
    SELECT 'OK - Sin nulos en clientes' AS Mensaje;

IF EXISTS (
    SELECT 1
    FROM stg_productos
    WHERE nombre_producto IS NULL OR categoria IS NULL OR precio_venta IS NULL
)
    SELECT 'Productos con campos nulos' AS Prueba, *
    FROM stg_productos
    WHERE nombre_producto IS NULL OR categoria IS NULL OR precio_venta IS NULL;
ELSE
    SELECT 'OK - Sin nulos en productos' AS Mensaje;

IF EXISTS (
    SELECT 1
    FROM stg_pedidos
    WHERE fecha_pedido IS NULL OR estado IS NULL OR id_cliente IS NULL
)
    SELECT 'Pedidos con campos nulos' AS Prueba, *
    FROM stg_pedidos
    WHERE fecha_pedido IS NULL OR estado IS NULL OR id_cliente IS NULL;
ELSE
    SELECT 'OK - Sin nulos en pedidos' AS Mensaje;
```

	Mensaje
1	OK - Sin nulos en clientes

	Mensaje
1	OK - Sin nulos en productos

	Mensaje
1	OK - Sin nulos en pedidos

3. Rangos de valores

Se validaron los rangos de fechas de pedidos (todas dentro del periodo esperado) y que los precios de los productos sean mayores a cero. Los resultados confirmaron la validez de estos campos.

```
IF EXISTS (
    SELECT 1 FROM stg_productos WHERE precio_venta <= 0
)
    SELECT 'Productos con precio <= 0' AS Prueba, *
    FROM stg_productos
    WHERE precio_venta <= 0;
ELSE
    SELECT 'OK - Precios válidos' AS Mensaje;

IF EXISTS (
    SELECT 1 FROM stg_productos WHERE cantidad_en_stock < 0
)
    SELECT 'Stock negativo' AS Prueba, *
    FROM stg_productos
    WHERE cantidad_en_stock < 0;
ELSE
    SELECT 'OK - Stock válido' AS Mensaje;

IF EXISTS (
    SELECT 1 FROM stg_clientes
    WHERE limite_credito < 0 OR limite_credito > 1000000
)
    SELECT 'Límites de crédito fuera de rango' AS Prueba, *
    FROM stg_clientes
    WHERE limite_credito < 0 OR limite_credito > 1000000;
ELSE
    SELECT 'OK - Límites de crédito válidos' AS Mensaje;
```

```
IF EXISTS (
    SELECT 1 FROM stg_detalle_pedido
    WHERE cantidad <= 0 OR precio_unidad <= 0
)
    SELECT 'Detalle con cantidad/precio inválido' AS Prueba, *
    FROM stg_detalle_pedido
    WHERE cantidad <= 0 OR precio_unidad <= 0;
ELSE
    SELECT 'OK - Cantidad y precio válidos en detalle' AS Mensaje;

IF EXISTS (
    SELECT 1 FROM stg_pedidos
    WHERE fecha_pedido < '2000-01-01' OR fecha_pedido > GETDATE()
)
    SELECT 'Pedidos con fecha fuera de rango' AS Prueba, *
    FROM stg_pedidos
    WHERE fecha_pedido < '2000-01-01' OR fecha_pedido > GETDATE();
ELSE
    SELECT 'OK - Fechas de pedidos válidas' AS Mensaje;
```

	Mensaje
1	OK - Precios válidos

	Mensaje
1	OK - Stock válido

	Mensaje
1	OK - Límites de crédito válidos

	Mensaje
1	OK - Cantidad y precio válidos en detalle

	Mensaje
1	OK - Fechas de pedidos válidas

Conclusión general de las validaciones.

En la actividad anterior (EA2), durante la creación de la base **Jardineria_Staging**, se identificaron y organizaron los problemas de calidad presentes en la base de origen, depurando los campos y registros necesarios. Como resultado de este proceso, la Staging actualmente cumple con todas las validaciones realizadas (conteos, huérfanos, duplicados, nulos y rangos), lo que confirma que está lista para ser utilizada en el proceso ETL.

Extracción de los datos desde la Base de Datos de Jardinería a la Base de Datos Jardinería_Staging

En la actividad anterior donde se trabajó la creación de la base de datos Jardineria_Staging se definió y ejecutó el proceso de extracción de datos desde la base de datos **Jardinería** hacia la base **Jardineria_Staging**, utilizando consultas SQL del tipo INSERT INTO ... SELECT .... Estas consultas permitieron trasladar únicamente los campos relevantes de cada tabla origen hacia sus correspondientes tablas en Staging, conforme al mapeo establecido. El detalle completo de los scripts utilizados se presenta en los **Anexos**, como evidencia del proceso realizado.

La verificación de la integridad y consistencia de los datos extraídos ya fue desarrollada en el apartado de *Validaciones de consistencia*. Allí se comprobó que los registros de la base **Jardineria_Staging** cumplen con los requisitos del modelo estrella, confirmando la correspondencia en los conteos entre origen y Staging, la ausencia de huérfanos, duplicados y nulos en columnas críticas, así como la validez de los rangos de fechas y precios.

Transformación de Datos con SQL Según las Necesidades Analíticas

En esta etapa se aplicaron técnicas de transformación de datos con el fin de preparar la información para el modelo estrella.

Durante la transformación se aplicaron las siguientes acciones:

- **Normalización de datos:** Separación de la fecha de pedido en día, mes, año y trimestre para facilitar el análisis temporal. Homogeneización de nombres de ciudad, país y región; eliminación de duplicados
- **Enriquecimiento:** concatenación de nombre y apellido de clientes y vendedores; categorización de productos; asignación de región a vendedores.
- **Cálculos de negocio:** creación de la métrica $Total = Cantidad \times Precio\_unidad$; cálculo de precio; generación de claves primarias para garantizar unicidad e integridad referencial.
- **Mapeo y validación de claves:** Generación de las claves primarias (ClienteID, ProductoID, VendedorID, TiempoID, VentaID) para asegurar unicidad. Así mismo, se realizará el mapeo correcto de las claves foráneas en la tabla de hechos hacia las dimensiones correspondientes, asegurando que cada referencia sea consistente. Finalmente, se llevará a cabo una verificación de la integridad referencial y de la consistencia de los datos, con el fin de mantener la calidad y confiabilidad del modelo estrella.

Tabla 2

DimCliente: Transformaciones y enriquecimientos

Staging	Transformación / enriquecimiento	Modelo estrella
nombre + apellido	Concatenar para formar NombreCliente	NombreCliente
ciudad	Limpiar mayúsculas/minúsculas, eliminar espacios	Ciudad
pais	Normalizar nombres	Pais
cliente_id	Validar único / crear si no existe	ClienteID
—	Eliminar duplicados	—

Nota. La tabla muestra las acciones necesarias para normalizar y enriquecer los datos de clientes desde la base staging hacia la dimensión DimCliente.

Tabla 3

DimProducto: Transformaciones y enriquecimientos

Staging	Transformación / enriquecimiento	Modelo estrella
producto_descripcion	Separar nombre y categoría	NombreProducto, Categoria
precio_unitario	Calcular Total / Cantidad si no existe	Precio
producto_id	Validar único / crear	ProductoID
—	Eliminar duplicados	—

Nota. La tabla muestra las acciones necesarias para normalizar y enriquecer los datos de producto desde la base staging hacia la dimensión DimProducto.

Tabla 4

DimVendedor: Transformaciones y enriquecimientos

Staging	Transformación / enriquecimiento	Modelo estrella
nombre + apellido	Concatenar para NombreVendedor	NombreVendedor
region_sucursal	Mapear región según sucursal	Region
vendedor_id	Validar único / crear	VendedorID
—	Eliminar duplicados	—

Nota. La tabla muestra las transformaciones aplicadas a los vendedores para preparar la dimensión DimVendedor

Tabla 5

DimTiempo: Transformaciones y enriquecimientos

Staging	Transformación / enriquecimiento	Modelo estrella
fecha_venta	Extraer año, mes, día, trimestre	Anio, Mes, Dia, Trimestre
fecha_venta	Formatear a YYYY-MM-DD	Fecha
—	Crear TiempoID único por fecha	TiempoID

Nota. La tabla muestra cómo procesar la fecha de venta para la dimensión temporal, facilitando análisis por periodos.

Tabla 6

FactVentas: Transformaciones y enriquecimientos

Staging	Transformación / enriquecimiento	Modelo estrella
—	Crear VentaID	VentaID
fecha_venta	Mapear a TiempoID	TiempoID
producto_id	Mapear a ProductoID	ProductoID
cliente_id	Mapear a ClienteID	ClienteID
vendedor_id	Mapear a VendedorID	VendedorID
cantidad	Validar >0	Cantidad
precio_unitario / total	Calcular Total = Cantidad × Precio	Total
—	Validar integridad referencial	—

Nota. La tabla detalla los pasos para garantizar integridad y calcular métricas en la tabla de hechos FactVentas.

La implementación de las transformaciones se realizó utilizando consultas SQL tipo INSERT INTO … SELECT, aplicando funciones de concatenación, conversión de fechas y cálculos aritméticos, con validaciones y filtros para garantizar la calidad y consistencia de los datos.

```sql
-- Clientes
INSERT INTO dbo.DimCliente (NombreCliente, Ciudad, Pais)
SELECT DISTINCT nombre_cliente, ciudad, pais
FROM Jardineria_Staging.dbo.stg_clientes;

-- Productos
INSERT INTO dbo.DimProducto (NombreProducto, Categoria, Precio)
SELECT nombre_producto, categoria, precio_venta
FROM Jardineria_Staging.dbo.stg_productos;

-- Vendedores
INSERT INTO dbo.DimVendedor (NombreVendedor, Region)
SELECT DISTINCT (e.nombre + ' ' + e.apellido), o.region
FROM Jardineria_Staging.dbo.stg_empleados e
JOIN Jardineria_Staging.dbo.stg_oficinas o ON e.id_oficina = o.id_oficina;

-- Tiempo
INSERT INTO dbo.DimTiempo (Fecha, Año, Mes, Dia, Trimestre)
SELECT DISTINCT
    fecha_pedido,
    YEAR(fecha_pedido),
    MONTH(fecha_pedido),
    DAY(fecha_pedido),
    DATEPART(QUARTER, fecha_pedido)
FROM Jardineria_Staging.dbo.stg_pedidos;
```

Carga de registros en el Data Mart final

Para cargar los datos transformados en el data mart final, se diseñaron consultas SQL de tipo **INSERT INTO … SELECT** que tomaron los registros limpios y enriquecidos desde la base de staging y los insertaron en las tablas de dimensiones y de hechos correspondientes. Posteriormente, se ejecutaron estas consultas y se realizaron verificaciones de consistencia e integridad, asegurando que todos los registros se insertaran correctamente y que las claves primarias y foráneas estuvieran correctamente mapeadas.

Script de Carga:

Se insertan los registros transformados desde la base de datos Jardimeria_Stagin al Modelo estrella

```
INSERT INTO dbo.FactVentas (ClienteID, ProductoID, VendedorID, TiempoID, Cantidad, Total)
SELECT
    dc.ClienteID,
    dp.ProductoID,
    dv.VendedorID,
    dt.TiempoID,
    dpe.cantidad,
    (dpe.cantidad * dpe.precio_unidad) AS Total
FROM Jardineria_Staging.dbo.stg_detalle_pedido dpe
JOIN Jardineria_Staging.dbo.stg_pedidos pe ON dpe.id_pedido = pe.id_pedido
JOIN Jardineria_Staging.dbo.stg_clientes c ON pe.id_cliente = c.id_cliente
JOIN Jardineria_Staging.dbo.stg_productos p ON dpe.id_producto = p.id_producto
JOIN DimCliente dc ON dc.ClienteID = c.id_cliente
JOIN DimProducto dp ON dp.NombreProducto = p.nombre_producto
JOIN DimTiempo dt ON dt.Fecha = pe.fecha_pedido
JOIN DimVendedor dv ON dv.VendedorID = (
        SELECT TOP 1 e.id_empleado
        FROM Jardineria_Staging.dbo.stg_empleados e
        WHERE e.id_empleado = (
            SELECT TOP 1 id_empleado
            FROM Jardineria_Staging.dbo.stg_empleados
        )
    );
```

Validaciones de Consistencia para el Data Mart Final:

Se verifica que los datos se hayan cargado correctamente al Data Mart final (modelo estrella)

1. **Conteo de Registros**

   Se comparó la cantidad de registros entre las tablas de staging y las tablas del Data Mart final para cada dimensión y la tabla de hechos. Los resultados mostraron que todos los registros esperados fueron insertados correctamente, asegurando que no se perdieron datos durante el proceso de carga.

```sql
-- Validación de Clientes
SELECT 'Clientes' AS Tabla,
       (SELECT COUNT(*) FROM Jardineria_Staging.dbo.stg_clientes) AS Registros_Staging,
       (SELECT COUNT(*) FROM dbo.DimCliente) AS Registros_Data_Mart_Final;

-- Validación de Productos
SELECT 'Productos' AS Tabla,
       (SELECT COUNT(*) FROM Jardineria_Staging.dbo.stg_productos) AS Registros_Staging,
       (SELECT COUNT(*) FROM dbo.DimProducto) AS Registros_Data_Mart_Final;

-- Validación de Vendedor
SELECT 'Empleados' AS Tabla,
       (SELECT COUNT(*) FROM Jardineria_Staging.dbo.stg_empleados) AS Registros_Staging,
       (SELECT COUNT(*) FROM dbo.DimVendedor) AS Registros_Data_Mart_Final;

-- Validación de Tiempo
SELECT 'Fecha Pedido' AS Tabla,
       (SELECT COUNT(DISTINCT CAST(fecha_pedido AS DATE)) FROM Jardineria_Staging.dbo.stg_pedidos) AS Fechas_Staging
       (SELECT COUNT(*) FROM dbo.DimTiempo) AS Registros_Data_Mart_Final;
```

Resultados | Mensajes

	Tabla	Registros_Staging	Registros_Data_Mart_Final
1	Clientes	36	36

	Tabla	Registros_Staging	Registros_Data_Mart_Final
1	Productos	276	276

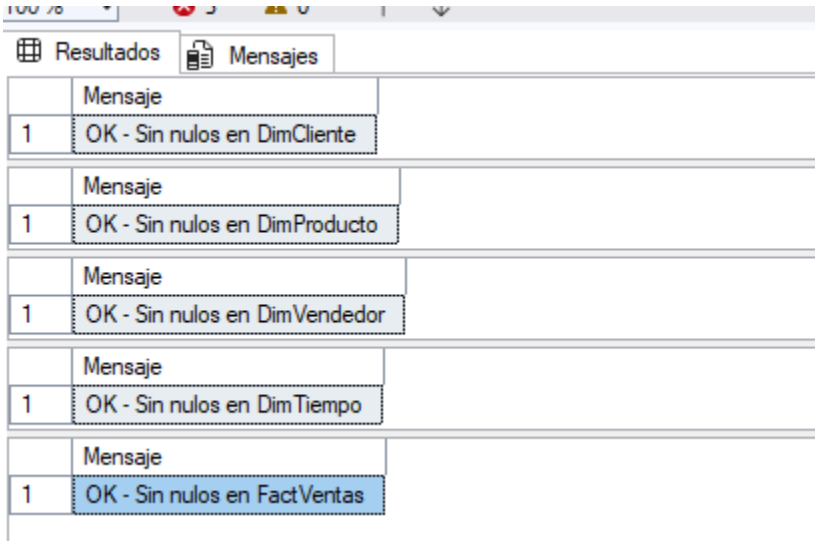
	Tabla	Registros_Staging	Registros_Data_Mart_Final
1	Empleados	31	31

	Tabla	Fechas_Staging	Registros_Data_Mart_Final
1	Fecha Pedido	78	78

2. Chequeo de Nulos

Se revisaron las claves primarias y foráneas en todas las tablas del Data Mart para detectar valores nulos. No se encontraron registros con claves vacías, lo que garantiza que cada registro está correctamente identificado y que las relaciones entre hechos y dimensiones son válidas.

```sql
-- DimCliente
IF EXISTS (SELECT 1 FROM dbo.DimCliente WHERE NombreCliente IS NULL OR Ciudad IS NULL OR Pais IS NULL)
    SELECT 'DimCliente con campos nulos' AS Prueba, *
    FROM dbo.DimCliente
    WHERE NombreCliente IS NULL OR Ciudad IS NULL OR Pais IS NULL;
ELSE
    SELECT 'OK - Sin nulos en DimCliente' AS Mensaje;

-- DimProducto
IF EXISTS (SELECT 1 FROM dbo.DimProducto WHERE NombreProducto IS NULL OR Categoria IS NULL OR Precio IS NULL)
    SELECT 'DimProducto con campos nulos' AS Prueba, *
    FROM dbo.DimProducto
    WHERE NombreProducto IS NULL OR Categoria IS NULL OR Precio IS NULL;
ELSE
    SELECT 'OK - Sin nulos en DimProducto' AS Mensaje;

-- DimVendedor
IF EXISTS (SELECT 1 FROM dbo.DimVendedor WHERE NombreVendedor IS NULL OR Region IS NULL)
    SELECT 'DimVendedor con campos nulos' AS Prueba, *
    FROM dbo.DimVendedor
    WHERE NombreVendedor IS NULL OR Region IS NULL;
ELSE
    SELECT 'OK - Sin nulos en DimVendedor' AS Mensaje;



-- DimTiempo
IF EXISTS (SELECT 1 FROM dbo.DimTiempo WHERE Fecha IS NULL)
    SELECT 'DimTiempo con Fecha nula' AS Prueba, *
    FROM dbo.DimTiempo
    WHERE Fecha IS NULL;
ELSE
    SELECT 'OK - Sin nulos en DimTiempo' AS Mensaje;

-- FactVentas
IF EXISTS (SELECT 1 FROM dbo.FactVentas
           WHERE ClienteID IS NULL OR ProductoID IS NULL OR VendedorID IS NULL
                 OR TiempoID IS NULL OR Cantidad IS NULL OR Total IS NULL)
    SELECT 'FactVentas con campos nulos' AS Prueba, *
    FROM dbo.FactVentas
    WHERE ClienteID IS NULL OR ProductoID IS NULL OR VendedorID IS NULL
          OR TiempoID IS NULL OR Cantidad IS NULL OR Total IS NULL;
ELSE
    SELECT 'OK - Sin nulos en FactVentas' AS Mensaje;
```



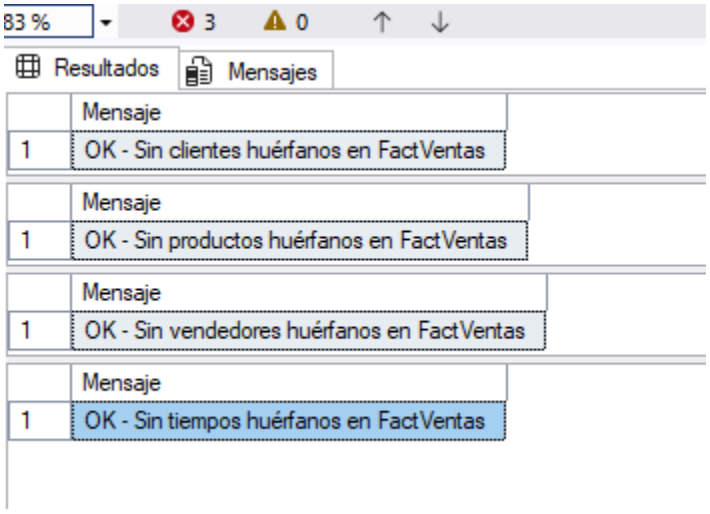
3. Integridad Referencial

Se comprobó que todas las claves foráneas de la tabla de hechos (FactVentas) existieran en las dimensiones correspondientes. La integridad referencial se cumplió en todos los casos, confirmando que los registros de la tabla de hechos están correctamente relacionados con sus dimensiones.

```sql
-- FactVentas con Cliente inexistente
IF EXISTS (
    SELECT 1
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimCliente c ON f.ClienteID = c.ClienteID
    WHERE c.ClienteID IS NULL
)
    SELECT 'FactVentas con Cliente inexistente' AS Prueba, f.VentaID, f.ClienteID
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimCliente c ON f.ClienteID = c.ClienteID
    WHERE c.ClienteID IS NULL;
ELSE
    SELECT 'OK - Sin clientes huérfanos en FactVentas' AS Mensaje;

-- FactVentas con Producto inexistente
IF EXISTS (
    SELECT 1
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimProducto p ON f.ProductoID = p.ProductoID
    WHERE p.ProductoID IS NULL
)
    SELECT 'FactVentas con Producto inexistente' AS Prueba, f.VentaID, f.ProductoID
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimProducto p ON f.ProductoID = p.ProductoID
    WHERE p.ProductoID IS NULL;
ELSE
    SELECT 'OK - Sin productos huérfanos en FactVentas' AS Mensaje;
```

```sql
-- FactVentas con Vendedor inexistente
IF EXISTS (
    SELECT 1
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimVendedor v ON f.VendedorID = v.VendedorID
    WHERE v.VendedorID IS NULL
)
    SELECT 'FactVentas con Vendedor inexistente' AS Prueba, f.VentaID, f.VendedorID
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimVendedor v ON f.VendedorID = v.VendedorID
    WHERE v.VendedorID IS NULL;
ELSE
    SELECT 'OK - Sin vendedores huérfanos en FactVentas' AS Mensaje;
```

```sql
-- FactVentas con Tiempo inexistente
IF EXISTS (
    SELECT 1
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimTiempo t ON f.TiempoID = t.TiempoID
    WHERE t.TiempoID IS NULL
)
    SELECT 'FactVentas con Tiempo inexistente' AS Prueba, f.VentaID, f.TiempoID
    FROM dbo.FactVentas f
    LEFT JOIN dbo.DimTiempo t ON f.TiempoID = t.TiempoID
    WHERE t.TiempoID IS NULL;
ELSE
    SELECT 'OK - Sin tiempos huérfanos en FactVentas' AS Mensaje;
```



4. Búsqueda de Duplicados

Se identificaron posibles registros duplicados en las dimensiones y no se encontraron coincidencias. Esto asegura que cada registro es único y que las dimensiones del modelo estrella no contienen información repetida que pueda afectar los análisis.

```sql
-- DimCliente
IF EXISTS (SELECT 1 FROM dbo.DimCliente GROUP BY NombreCliente, Ciudad, Pais HAVING COUNT(*) > 1)
    SELECT 'DimCliente duplicados' AS Prueba, NombreCliente, Ciudad, Pais, COUNT(*) AS Total
    FROM dbo.DimCliente
    GROUP BY NombreCliente, Ciudad, Pais
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin duplicados en DimCliente' AS Mensaje;

-- DimProducto
IF EXISTS (SELECT 1 FROM dbo.DimProducto GROUP BY NombreProducto, Categoria, Precio HAVING COUNT(*) > 1)
    SELECT 'DimProducto duplicados' AS Prueba, NombreProducto, Categoria, Precio, COUNT(*) AS Total
    FROM dbo.DimProducto
    GROUP BY NombreProducto, Categoria, Precio
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin duplicados en DimProducto' AS Mensaje;

-- DimVendedor
IF EXISTS (SELECT 1 FROM dbo.DimVendedor GROUP BY NombreVendedor, Region HAVING COUNT(*) > 1)
    SELECT 'DimVendedor duplicados' AS Prueba, NombreVendedor, Region, COUNT(*) AS Total
    FROM dbo.DimVendedor
    GROUP BY NombreVendedor, Region
    HAVING COUNT(*) > 1;
ELSE
    SELECT 'OK - Sin duplicados en DimVendedor' AS Mensaje;
```

Resultados | Mensajes

	Prueba	NombreCliente	Ciudad	Pais	Total
1	DimCliente duplicados	Lasas S.A.	Fuenlabrada	Spain	2

	Mensaje
1	OK - Sin duplicados en DimProducto

	Mensaje
1	OK - Sin duplicados en DimVendedor

Consultas y Resultados

Después de completar todo el proceso ETL y la carga al Data Mart final, se ejecutaron las consultas correspondientes para obtener la información solicitada en las tres preguntas planteadas: producto más vendido, categoría con más productos y año con mayores ventas, obteniéndose los resultados esperados.

```sql
SELECT TOP 1 dp.NombreProducto, SUM(fv.Cantidad) AS TotalUnidadesVendidas
FROM dbo.FactVentas fv
JOIN dbo.DimProducto dp ON fv.ProductoID = dp.ProductoID
GROUP BY dp.NombreProducto
ORDER BY TotalUnidadesVendidas DESC;

-- Categoría con más productos
SELECT TOP 1 Categoria, COUNT(*) AS CantidadProductos
FROM dbo.DimProducto
GROUP BY Categoria
ORDER BY CantidadProductos DESC;

-- Año con mayores ventas
SELECT TOP 1 dt.Año, SUM(fv.Total) AS TotalVentas
FROM dbo.FactVentas fv
JOIN dbo.DimTiempo dt ON fv.TiempoID = dt.TiempoID
GROUP BY dt.Año
ORDER BY TotalVentas DESC;
```

Resultados | Mensajes

	NombreProducto	TotalUnidadesVendidas
1	Cerezo	2212

	Categoria	CantidadProductos
1	Ornamentales	154

	Año	TotalVentas
1	2008	281827.00

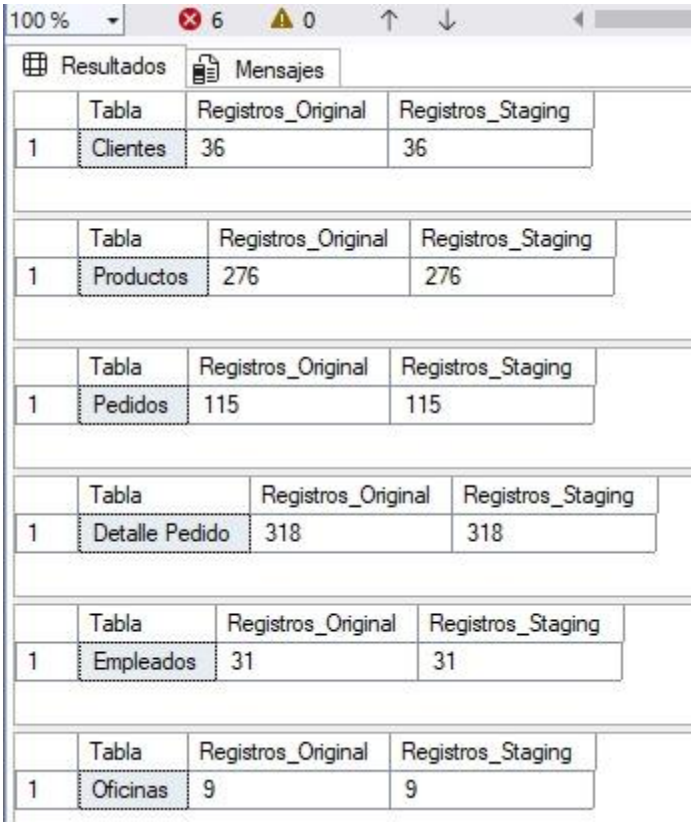
Anexos

En esta sección se incluyen las capturas de pantalla generadas durante el desarrollo del proyecto, con el fin de evidenciar los procesos realizados. Se incluyen los respaldos (*backups*) de las bases de datos empleadas, así como el documento con los scripts del proceso ETL y carga al Data Mart final. Estos anexos permiten garantizar la trazabilidad, la reproducibilidad del proceso y la verificación de la estructura propuesta en el modelo de datos.

- Validaciones Tablas Jardinería y Jardinería_Staging:

```sql
-- Validación de Clientes
SELECT 'Clientes' AS Tabla,
       (SELECT COUNT(*) FROM dbo.cliente) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_Cliente) AS Registros_Staging;

-- Validación de Productos
SELECT 'Productos' AS Tabla,
       (SELECT COUNT(*) FROM dbo.producto) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_Producto) AS Registros_Staging;

-- Validación de Pedidos
SELECT 'Pedidos' AS Tabla,
       (SELECT COUNT(*) FROM dbo.pedido) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_Pedido) AS Registros_Staging;

-- Validación de Detalle de Pedidos
SELECT 'Detalle Pedido' AS Tabla,
       (SELECT COUNT(*) FROM dbo.detalle_pedido) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_DetallePedido) AS Registros_Staging;

-- Validación de Empleados
SELECT 'Empleados' AS Tabla,
       (SELECT COUNT(*) FROM dbo.empleado) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_Empleado) AS Registros_Staging;

-- Validación de Oficinas
SELECT 'Oficinas' AS Tabla,
       (SELECT COUNT(*) FROM dbo.oficina) AS Registros_Original,
       (SELECT COUNT(*) FROM dbo.Stg_Oficina) AS Registros_Staging;
```

- Resultados obtenidos a las Validaciones:



100 % 6 0

Resultados Mensajes

	Tabla	Registros_Original	Registros_Staging
1	Clientes	36	36

	Tabla	Registros_Original	Registros_Staging
1	Productos	276	276

	Tabla	Registros_Original	Registros_Staging
1	Pedidos	115	115

	Tabla	Registros_Original	Registros_Staging
1	Detalle Pedido	318	318

	Tabla	Registros_Original	Registros_Staging
1	Empleados	31	31

	Tabla	Registros_Original	Registros_Staging
1	Oficinas	9	9

- Extracción desde la BD Jardinería a la BD Jardinería _Staging:

```
        -- Poblar Stg_Cliente
INSERT INTO dbo.Stg_Cliente (id_cliente, nombre_cliente, ciudad, pais, limite_credito)
SELECT
    ID_cliente,
    nombre_cliente,
    ciudad,
    pais,
    limite_credito
FROM dbo.cliente;

-- Poblar Stg_Producto
INSERT INTO dbo.Stg_Producto (id_producto, nombre_producto, categoria, precio_venta)
SELECT
    ID_producto,
    nombre,
    c.Desc_Categoria,
    precio_venta
FROM dbo.producto p
JOIN dbo.Categoria_producto c ON c.Id_Categoria = p.Categoria;
```

```sql
-- Poblar Stg_Pedido
INSERT INTO dbo.Stg_Pedido (id_pedido, fecha_pedido, estado, id_cliente)
SELECT
    ID_pedido,
    fecha_pedido,
    estado,
    ID_cliente
FROM dbo.pedido;

-- Poblar Stg_DetallePedido
INSERT INTO dbo.Stg_DetallePedido (id_pedido, id_producto, cantidad, precio_unidad)
SELECT
    ID_pedido,
    ID_producto,
    cantidad,
    precio_unidad
FROM dbo.detalle_pedido;

-- Poblar Stg_Empleado
INSERT INTO dbo.Stg_Empleado (id_empleado, nombre, apellido1, puesto, id_oficina)
SELECT
    ID_empleado,
    nombre,
    apellido1,
    puesto,
    ID_oficina
FROM dbo.empleado;
```

```sql
-- Poblar Stg_Oficina
INSERT INTO dbo.Stg_Oficina (id_oficina, ciudad, pais, region)
SELECT
    ID_oficina,
    ciudad,
    pais,
    region
FROM dbo.oficina;
```

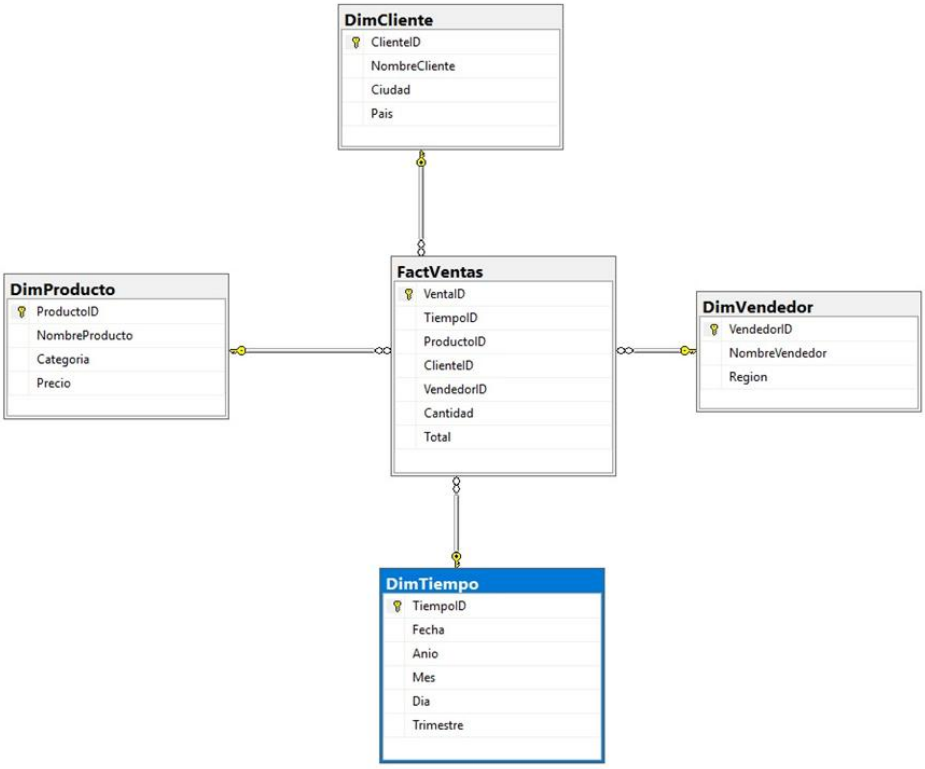
- Link a Backups de ambas Bases de datos:

  Link BK Jardinería

  Link BK Jardienría-Staging

  Link BK Jardinería_Data_Mart_Final

- Data Mart Final (modelo estrella).



Conclusiones

- **Objetivo 1:** Verificar la disponibilidad y consistencia de la base de datos Staging previamente creada, asegurando que los datos estén listos para el proceso ETL.

  Conclusión: La revisión confirmó que la base Staging estaba disponible y consistente, garantizando que los datos podían ser procesados correctamente en las siguientes etapas del ETL.
- **Objetivo 2:** Transformar y depurar los datos en Staging, aplicando validaciones y cálculos necesarios para asegurar su calidad y consistencia.

  Conclusión: Se aplicaron transformaciones, depuraciones y validaciones, asegurando que los datos fueran limpios, consistentes y listos para su carga en el Data Mart.
- **Objetivo 3:** Cargar los datos transformados en el Data Mart con modelo estrella, garantizando la correcta integración de las dimensiones y la tabla de hechos.

  Conclusión: Los datos fueron cargados correctamente en el Data Mart, verificando la integridad de las claves primarias y foráneas y la correcta relación entre dimensiones y hechos.
- **Objetivo 4:** Validar los resultados mediante consultas analíticas que permitan responder preguntas clave del negocio, como el producto más vendido, la categoría con más productos y los años con mayores ventas.

  Conclusión: Se ejecutaron consultas analíticas que confirmaron la correcta carga y transformación de los datos, permitiendo responder satisfactoriamente las preguntas de negocio planteadas.

Bibliografía

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3.ª ed.). Wiley.

Inmon, W. H. (2005). *Building the data warehouse* (4.ª ed.). Wiley.

Microsoft. (s. f.). *Backup and restore of SQL Server databases*. Microsoft Docs. Recuperado de https://learn.microsoft.com/sql/backup-restore/backup-and-restore-sql-server-databases