

**Evidencia de aprendizaje 3. Proceso de transformación de datos y carga
en el data mart final**

Sebastián Castaño Cossio

IU DIGITAL DE ANTIOQUIA

**Docente:
Víctor Mercado**

**Bases de Datos II
PREICA2402B010070**

8 Octubre 2024

Introducción

El modelo estrella es una de las arquitecturas de datos más comunes y eficaces para la creación de Data Marts en entornos de Business Intelligence (BI). Este modelo facilita el análisis de grandes volúmenes de datos y permite responder a preguntas de negocio específicas con rapidez y eficiencia. En este caso, utilizaremos la base de datos "Jardinería" para construir un modelo estrella que permita analizar las ventas y responder preguntas clave sobre productos, categorías y tendencias anuales de ventas.

Objetivos

El objetivo principal de este modelo estrella es proporcionar un esquema que permita:

1. Identificar el producto más vendido.
2. Determinar la categoría con más productos.
3. Identificar el año con más ventas.

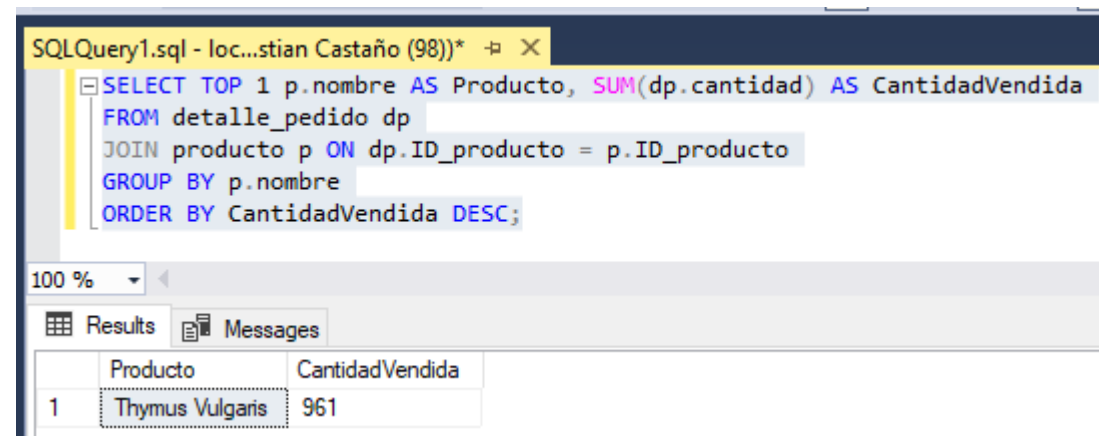
Planteamiento del Problema

El análisis de las ventas es importante para cualquier empresa, permitiendo identificar productos exitosos y entender las preferencias de los clientes. Sin embargo, sin una estructura de datos bien definida, obtener esta información puede ser difícil. El objetivo es crear un modelo estrella basado en la base de datos "Jardinería" para facilitar este análisis.

Análisis del Problema

La base de datos "Jardinería" contiene tablas que registran información sobre productos, clientes, pedidos y pagos. La clave para construir el modelo estrella es identificar las relaciones entre estas tablas y estructurarlas de tal manera que se puedan realizar análisis eficientes y precisos.

1. Identificar el producto más vendido.



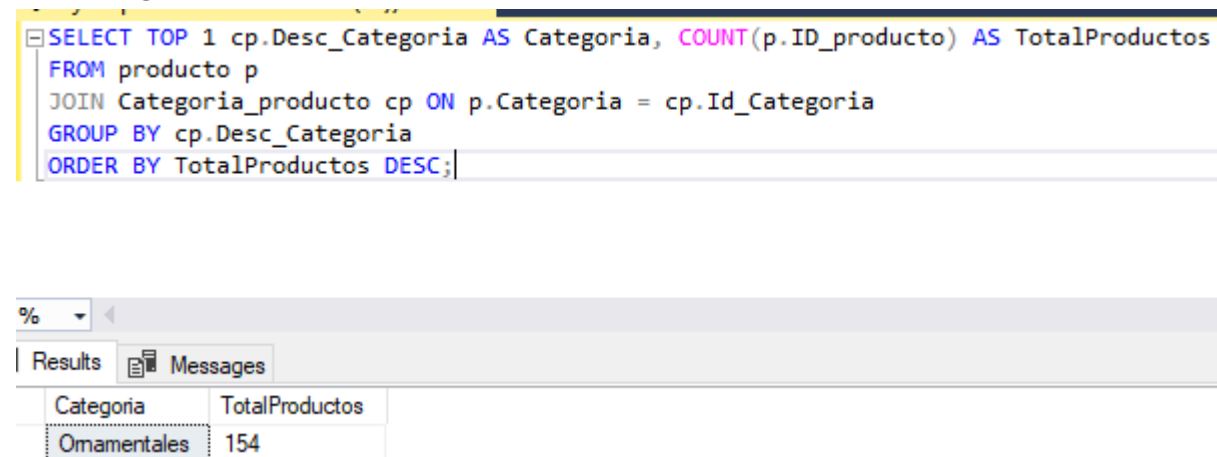
The screenshot shows a SQL query window titled "SQLQuery1.sql - loc...stian Castaño (98)*". The query is as follows:

```
SELECT TOP 1 p.nombre AS Producto, SUM(dp.cantidad) AS CantidadVendida
FROM detalle_pedido dp
JOIN producto p ON dp.ID_producto = p.ID_producto
GROUP BY p.nombre
ORDER BY CantidadVendida DESC;
```

Below the query, the "Results" tab is active, displaying a single row of data:

	Producto	CantidadVendida
1	Thymus Vulgaris	961

2. La categoría con más productos:



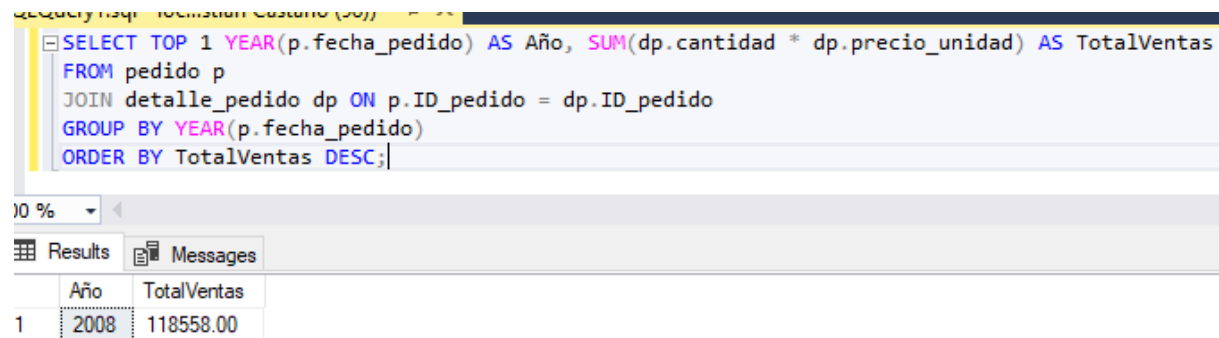
The screenshot shows a SQL query window with the following query:

```
SELECT TOP 1 cp.Desc_Categoria AS Categoria, COUNT(p.ID_producto) AS TotalProductos
FROM producto p
JOIN Categoria_producto cp ON p.Categoria = cp.Id_Categoria
GROUP BY cp.Desc_Categoria
ORDER BY TotalProductos DESC;
```

Below the query, the "Results" tab is active, displaying a single row of data:

	Categoria	TotalProductos
	Ornamentales	154

3. El año con más ventas:



The screenshot shows a SQL query in the 'Query Editor' window of SQL Server Enterprise Manager. The query is designed to find the year with the highest total sales. Below the query editor, the 'Results' pane displays a single row of data.

```
SELECT TOP 1 YEAR(p.fecha_pedido) AS Año, SUM(dp.cantidad * dp.precio_unidad) AS TotalVentas
FROM pedido p
JOIN detalle_pedido dp ON p.ID_pedido = dp.ID_pedido
GROUP BY YEAR(p.fecha_pedido)
ORDER BY TotalVentas DESC;
```

	Año	TotalVentas
1	2008	118558.00

Diseño del modelo estrella.

1. Identificación de la tabla de hechos:

la tabla de hechos sería detalle_pedido, ya que contiene los datos de las ventas, como la cantidad de productos vendidos, el precio por unidad y el ID del producto, que son claves para realizar análisis de ventas.

2. Identificación de las dimensiones:

Las dimensiones son aquellas tablas que contienen información que ayuda a contextualizar los hechos. Las dimensiones que podríamos diseñar son:

- Dimensión Producto: Información sobre cada producto.
- Dimensión Categoría de Producto: Información sobre la categoría de cada producto.
- Dimensión Cliente: Información sobre los clientes.
- Dimensión Empleado (Representante de ventas): Información sobre los empleados que gestionan las ventas.
- Dimensión Tiempo: Descripción de las fechas de los pedidos (fecha de pedido, fecha esperada, etc.).

3. Modelo estrella

Tabla de hechos: Hechos_Ventas

Esta tabla contiene los detalles de las ventas (cada venta representa una fila en la tabla de hechos):

- **ID_venta (INT):** Clave primaria.
- **ID_producto (INT):** Clave foránea a **Dim_Producto**.
- **ID_cliente (INT):** Clave foránea a **Dim_Cliente**.
- **ID_empleado (INT):** Clave foránea a **Dim_Empleado**.
- **ID_tiempo (INT):** Clave foránea a **Dim_Tiempo**.
- **Cantidad (INT):** Cantidad de productos vendidos.

- **Precio_unidad (NUMERIC(15,2))**: Precio por unidad.
- **Total_venta (NUMERIC(15,2))**: Total de la venta (Cantidad * Precio_unidad).

Dimensiones:

Dim_Producto:

- **ID_producto (INT)**: Clave primaria.
- **Codigo_producto (VARCHAR(15))**: Código del producto.
- **Nombre (VARCHAR(70))**: Nombre del producto.
- **Categoria (VARCHAR(50))**: Descripción de la categoría.
- **Proveedor (VARCHAR(50))**: Proveedor del producto.
- **Descripcion (TEXT)**: Descripción detallada del producto.
- **Cantidad_en_stock (SMALLINT)**: Cantidad disponible en inventario.
- **Precio_venta (NUMERIC(15,2))**: Precio de venta.
- **Precio_proveedor (NUMERIC(15,2))**: Precio al que se compra al proveedor

Dim_Cliente:

- **ID_cliente (INT)**: Clave primaria.
- **Nombre_cliente (VARCHAR(50))**: Nombre de la empresa o cliente.
- **Nombre_contacto (VARCHAR(30))**: Nombre del contacto.
- **Apellido_contacto (VARCHAR(30))**: Apellido del contacto.
- **Telefono (VARCHAR(15))**: Número de teléfono del cliente.
- **Fax (VARCHAR(15))**: Número de fax.
- **Direccion_1 (VARCHAR(50))**: Línea de dirección principal.
- **Direccion_2 (VARCHAR(50))**: Línea de dirección secundaria.
- **Ciudad (VARCHAR(50))**: Ciudad del cliente.
- **Region (VARCHAR(50))**: Región.
- **Pais (VARCHAR(50))**: País del cliente.
- **Codigo_postal (VARCHAR(10))**: Código postal.
- **Limite_credito (NUMERIC(15, 2))**: Límite de crédito del cliente.

Dim_Empleado:

- **ID_empleado (INT)**: Clave primaria.
- **Nombre (VARCHAR(50))**: Nombre del empleado.
- **Apellido (VARCHAR(50))**: Apellido del empleado.
- **Email (VARCHAR(100))**: Correo electrónico.
- **Puesto (VARCHAR(50))**: Puesto del empleado.

Dim_Tiempo:

- **ID_tiempo (INT): Clave primaria.**
- **Fecha (DATE): Fecha completa.**
- **Año (INT): Año.**
- **Mes (INT): Mes.**
- **Día (INT): Día.**
- **Trimestre (INT): Trimestre del año.**
- **Semana_del_año (INT): Número de la semana.**
- **Semestre (INT): Semestre del año.**
- **Bimestre (INT): Bimestre del año.**
- **Día_semana (INT): Día de la semana.**
- **Es_festivo (BIT): Indica si es día festivo**

4. Relaciones del modelo estrella

La tabla de hechos **Hechos_Ventas** se relaciona con las dimensiones a través de las claves foráneas (FK). Las relaciones entre las tablas serían las siguientes:

Hechos_Ventas(ID_producto) → Dim_Producto(ID_producto)

Hechos_Ventas(ID_cliente) → Dim_Cliente(ID_cliente)

Hechos_Ventas(ID_empleado) → Dim_Empleado(ID_empleado)

Hechos_Ventas(ID_tiempo) → Dim_Tiempo(ID_tiempo)

Evidencia de aprendizaje 2. Creación de una base de datos de Staging

El primer paso es revisar la estructura de la base de datos "Jardinería". Permíteme revisar el archivo SQL que adjuntaste para identificar las tablas y columnas que contiene y así decidir qué datos son relevantes para trasladar a la base de datos **Staging**.

1. **Oficina:** Almacena datos sobre las oficinas, con información como descripción, ciudad, país, y dirección.
2. **Empleado:** Almacena información de los empleados, como nombre, apellidos, oficina asociada y jefe directo.
3. **Categoría de Producto:** Contiene datos sobre las categorías de productos, incluyendo descripciones y posibles imágenes.
4. **Cliente:** Información sobre los clientes, como nombre del cliente, información de contacto y límites de crédito.
5. **Pedido:** Registros de pedidos, incluyendo fechas de pedido, fecha esperada, estado y cliente relacionado.

Podemos mover las siguientes tablas relevantes a la **base de datos Staging** para transformaciones futuras:

- **Oficina**
- **Empleado**
- **Cliente**
- **Pedido**
- **Producto**
- **Detalle_pedido**
- **Pago**

2. Construcción de la base de Datos Staging:

Estructura de las tablas en Staging:

Crear las tablas en **Staging** replicando la estructura de las tablas más importantes de "Jardinería", pero sin relaciones estrictas (como claves foráneas), ya que el objetivo de Staging es realizar transformaciones antes de mover los datos a una base de datos final.

- **Stg_Oficina:**
 - Replicación de la tabla oficina.
- **Stg_Empleado:**
 - Replicación de la tabla empleado, pero sin las restricciones de claves foráneas.
- **Stg_Cliente:**
 - Replicación de la tabla cliente.
- **Stg_Pedido:**
 - Replicación de la tabla pedido.

Consultas para traer los datos de Jardinería a Staging:

1. Consulta para traer los datos de oficina:

```
INSERT INTO Stg_Oficina (ID_oficina, Descripcion, ciudad,
pais, region, codigo_postal, telefono, linea_direccion1,
linea_direccion2)
SELECT ID_oficina, Descripcion, ciudad, pais, region,
codigo_postal, telefono, linea_direccion1, linea_direccion2
FROM oficina;
```

2. Consulta para traer los datos de empleado:

```
INSERT INTO Stg_Empleado (ID_empleado, nombre, apellido1,
apellido2, extension, email, ID_oficina, ID_jefe,
puesto)SELECT ID_empleado, nombre, apellido1, apellido2,
extension, email, ID_oficina, ID_jefe, puesto
```

```
FROM empleado;
```

3. **Consulta para traer los datos de cliente:**

```
INSERT INTO Stg_Cliente (ID_cliente, nombre_cliente,  
nombre_contacto, apellido_contacto, telefono, fax,  
linea_direccion1, linea_direccion2, ciudad, region, pais,  
codigo_postal, ID_empleado_rep_ventas, limite_credito)SELECT  
ID_cliente, nombre_cliente, nombre_contacto,  
apellido_contacto, telefono, fax, linea_direccion1,  
linea_direccion2, ciudad, region, pais, codigo_postal,  
ID_empleado_rep_ventas, limite_credito
```

```
FROM cliente;
```

4. **Consulta para traer los datos de pedidos:**

```
INSERT INTO Stg_Pedido (ID_pedido, fecha_pedido,  
fecha_esperada, fecha_entrega, estado, comentarios,  
ID_cliente)SELECT ID_pedido, fecha_pedido, fecha_esperada,  
fecha_entrega, estado, comentarios, ID_cliente
```

```
FROM pedido;
```

Paso 1: Crear la Base de Datos Staging

Primero, debemos crear la base de datos **Staging**.

```
CREATE DATABASE staging;
```

Paso 2: Crear las Tablas en Staging

tablas que replicarán la estructura de **Jardinería**, pero sin restricciones (claves foráneas).

Crear tabla Stg_Oficina:

```
CREATE TABLE Stg_Oficina (  
ID_oficina INT,  
Descripcion VARCHAR(10),  
ciudad VARCHAR(30),  
pais VARCHAR(50),  
region VARCHAR(50),  
codigo_postal VARCHAR(10),  
telefono VARCHAR(20),  
linea_direccion1 VARCHAR(50),  
linea_direccion2 VARCHAR(50)
```



```
);
```

Crear tabla Stg_Empleado:

```
CREATE TABLE Stg_Empleado (  
    ID_empleado INT,  
    nombre VARCHAR(50),  
    apellido1 VARCHAR(50),  
    apellido2 VARCHAR(50),  
    extension VARCHAR(10),  
    email VARCHAR(100),  
    ID_oficina INT,  
    ID_jefe INT,  
    puesto VARCHAR(50)  
);
```

Crear tabla Stg_Cliente:

```
CREATE TABLE Stg_Cliente (  
    ID_cliente INT,  
    nombre_cliente VARCHAR(50),  
    nombre_contacto VARCHAR(30),  
    apellido_contacto VARCHAR(30),  
    telefono VARCHAR(15),  
    fax VARCHAR(15),  
    linea_direccion1 VARCHAR(50),  
    linea_direccion2 VARCHAR(50),  
    ciudad VARCHAR(50),  
    region VARCHAR(50),  
    pais VARCHAR(50),  
    codigo_postal VARCHAR(10),  
    ID_empleado_rep_ventas INT,  
    limite_credito NUMERIC(15, 2)  
);
```

Crear tabla Stg_Pedido:

```
CREATE TABLE Stg_Pedido (  
    ID_pedido INT,  
    fecha_pedido DATE,  
    fecha_esperada DATE,  
    fecha_entrega DATE,  
    estado VARCHAR(15),  
    comentarios TEXT,
```

```
ID_cliente INT  
);
```

Paso 3: Insertar los Datos desde Jardinería a Staging

consultas para trasladar los datos desde **Jardinería** a las tablas de **Staging**.

Insertar datos en Stg_Oficina:

```
INSERT INTO Stg_Oficina (ID_oficina, Descripcion, ciudad, pais,  
region, codigo_postal, telefono, linea_direccion1, linea_direccion2)  
SELECT ID_oficina, Descripcion, ciudad, pais, region, codigo_postal,  
telefono, linea_direccion1, linea_direccion2  
FROM jardineria.dbo.oficina;
```

Insertar datos en Stg_Empleado:

```
INSERT INTO Stg_Empleado (ID_empleado, nombre, apellido1, apellido2,  
extension, email, ID_oficina, ID_jefe, puesto)  
SELECT ID_empleado, nombre, apellido1, apellido2, extension, email,  
ID_oficina, ID_jefe, puesto  
FROM jardineria.dbo.empleado;
```

Insertar datos en Stg_Cliente:

```
INSERT INTO Stg_Cliente (ID_cliente, nombre_cliente,  
nombre_contacto, apellido_contacto, telefono, fax, linea_direccion1,  
linea_direccion2, ciudad, region, pais, codigo_postal,  
ID_empleado_rep_ventas, limite_credito)  
SELECT ID_cliente, nombre_cliente, nombre_contacto,  
apellido_contacto, telefono, fax, linea_direccion1,  
linea_direccion2, ciudad, region, pais, codigo_postal,  
ID_empleado_rep_ventas, limite_credito  
FROM jardineria.dbo.cliente;
```

Insertar datos en Stg_Pedido:

```
INSERT INTO Stg_Pedido (ID_pedido, fecha_pedido, fecha_esperada,  
fecha_entrega, estado, comentarios, ID_cliente)  
SELECT ID_pedido, fecha_pedido, fecha_esperada, fecha_entrega,  
estado, comentarios, ID_cliente  
FROM jardineria.dbo.pedido;
```

Paso 4: Validar los Datos en Staging

Después de ejecutar estas consultas, validar que los datos hayan sido insertados correctamente ejecutando:

```
SELECT * FROM Stg_Oficina;
```

```
SELECT * FROM Stg_Empleado;  
SELECT * FROM Stg_Cliente;  
SELECT * FROM Stg_Pedido;
```

Paso 5: Crear los Backups de ambas Bases de Datos

Una vez que los datos han sido trasladados correctamente, se procede a crear los backups de las bases de datos **Jardinería** y **Staging**.

Crear Backup de Jardinería:

```
BACKUP DATABASE jardineria  
TO DISK = 'C:\backup\jardineria.bak';
```

Crear Backup de Staging:

```
BACKUP DATABASE staging  
TO DISK = 'C:\backup\staging.bak';
```

Limpieza y normalización de datos

Vamos a asegurarnos de que los datos en las tablas replicadas en la base de datos staging sean consistentes y completos. Algunas de las acciones clave incluyen:

Eliminar duplicados: Garantizar que no haya filas duplicadas en las tablas de staging.

Manejo de valores nulos: Identificar columnas con valores nulos y definir un tratamiento adecuado, como reemplazarlos con valores por defecto o eliminarlos si es necesario.

Normalización de datos: Asegurarnos de que los datos como nombres y direcciones estén en un formato consistente (uso de mayúsculas, espacios correctos, etc.).

Ejemplo de consulta para eliminar duplicados:

```
DELETE FROM Stg_Cliente  
WHERE ID_cliente IN (  
    SELECT ID_cliente  
    FROM (  
        SELECT ID_cliente, ROW_NUMBER() OVER(PARTITION BY ID_cliente  
ORDER BY ID_cliente) AS row_num  
        FROM Stg_Cliente  
    ) t  
    WHERE t.row_num > 1  
);
```

Enriquecimiento de datos

En esta etapa, se pueden añadir más datos relevantes para mejorar los análisis. Por ejemplo, podríamos crear nuevas columnas en la tabla de tiempo para almacenar datos como el trimestre, el semestre o identificar si es un día festivo.

Ejemplo de consulta para enriquecer la dimensión de tiempo:

```
UPDATE Stg_Tiempo SET trimestre = CASE WHEN mes BETWEEN 1 AND 3 THEN  
1 WHEN mes BETWEEN 4 AND 6 THEN 2 WHEN mes BETWEEN 7 AND 9 THEN 3  
ELSE 4 END, semestre = CASE WHEN mes BETWEEN 1 AND 6 THEN 1 ELSE 2  
END;
```

Transformación de datos para garantizar la coherencia

Se deben realizar las transformaciones necesarias utilizando herramientas ETL o directamente en SQL para asegurar que los datos tengan integridad referencial y estén listos para su análisis.

Normalización de fechas: Asegurarnos de que todas las fechas tengan un formato coherente en las tablas.

Conversión de tipos de datos: Convertir valores numéricos o de texto cuando sea necesario.

Ejemplo de conversión de tipo de dato en precios:

```
UPDATE Stg_Producto SET precio_venta = CONVERT(DECIMAL(10, 2),  
precio_venta) WHERE precio_venta IS NOT NULL;
```

Carga en el Data Mart final

Tras la limpieza, normalización y transformación de los datos, podemos cargarlos en el Data Mart final.

Carga de la tabla de hechos (Hechos_Ventas):

```
INSERT INTO Hechos_Ventas (ID_producto, ID_cliente, ID_empleado,  
ID_tiempo, Cantidad, Precio_unidad, Total_venta)  
SELECT dp.ID_producto, p.ID_cliente, e.ID_empleado, t.ID_tiempo,  
dp.cantidad, dp.precio_unidad, (dp.cantidad * dp.precio_unidad)  
FROM Stg_Detalle_Pedido dp  
JOIN Stg_Pedido p ON dp.ID_pedido = p.ID_pedido  
JOIN Stg_Empleado e ON p.ID_empleado_rep_ventas = e.ID_empleado  
JOIN Stg_Tiempo t ON p.fecha_pedido = t.fecha;
```

Validaciones finales

Antes de dar por finalizado el proceso, se debe validar que los datos se hayan cargado correctamente en el Data Mart final.

Verificar consistencia en la tabla de hechos:

```
SELECT COUNT(*) AS Total_Registros, SUM(Total_venta) AS  
Ventas_Totales FROM Hechos_Ventas;
```

Repositorio:

https://github.com/scastano21/data_mart_jardineria.git