

F20 - EVIDENCIA DE APRENDIZAJE 4. PROYECTO INTEGRADOR-REPOSITORIO DE TODAS LAS ACTIVIDADES

JUAN CAMILO CASTELLANOS RODRÍGUEZ
PREICA2401B010093
INGENIERÍA DE SOFTWARE Y DATOS

DOCENTE
VICTOR HUGO MERCADO
BASES DE DATOS II

BOGOTÁ, 23 DE MAYO DE 2024
INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

Contenido

Introducción	4
Objetivos	5
Planteamiento del problema (Modelo estrella).....	6
Análisis del problema (Modelo estrella)	6
Propuesta de solución (modelo estrella)	7
Dimensiones.....	7
Diseño del modelo estrella.....	8
Query - creación modelo estrella jardinería	8
Propuesta de solución (Staging).....	10
Staging en visual studio.....	11
Limpiar tablas: truncate	11
Staging en SQL Server.....	12
Querys para generar Staging.....	13
ELEMENTOS USADOS PARA LA TRANSFORMACIÓN DE DATOS.....	17
<i>Data Flow Task</i>	17
Origen de ADO NET	17
Destino De ADO NET	17
Sort	17
Merge Join.....	17
Conditional Split	17
Transformación Tabla tiempo	17
Tabla de destino tiempo	18
Transformación tabla cliente	19
Tabla de destino cliente	19
Transformación tabla empleado	20
Tabla de destino empleado.....	20
Transformación tabla producto	21
Tabla de destino producto	21
Transformación tabla FAC	22
Tabla de destino FAC.....	22
Querys usados para la transformación	23
CARGA A DATAMART	25

Carga de la tabla Tiempo a Datamart (Carga_Tiempo_DM).....	25
Resultado Carga Tiempo a Datamart	26
Carga de la tabla Cliente a Datamart (Carga_Cliente_DM).....	27
Resultado Carga Cliente a Datamart	28
Carga de la tabla Empleado a Datamart (Carga_Emppleado_DM).....	29
Resultado Carga Empleado a Datamart	30
Carga de la tabla Producto a Datamart (Carga_Producto_DM).....	31
Resultado Carga Producto a Datamart.....	32
Carga de la tabla Hechos FAC a Datamart (Carga_Hechos_FAC_DM)	33
Resultado Carga Hechos FAC a Datamart	34
Querys usados para la carga en DM.....	35
ETL Final.....	37
Ejecución Final ETL (OK)	37
Modelo Estrella Final BD: DMVentas_Jardineria	38
LINK REPOSITORIO	38
Conclusiones	39
Bibliografía	40

Introducción

El proceso de Extracción, Transformación y Carga (ETL) es crucial para la gestión de datos en las organizaciones actuales. Este proceso permite recolectar datos de diversas fuentes, transformarlos para adaptarlos a las necesidades analíticas y cargarlos en una base de datos centralizada. Esto facilita la toma de decisiones basada en información precisa y actualizada. Aprender a implementar una ETL no solo mejora la eficiencia operativa, sino que también potencia la capacidad de análisis y la competitividad de una empresa. En este trabajo, se describen los pasos seguidos para desarrollar un proceso ETL, destacando la importancia de cada etapa y su contribución al manejo eficiente de los datos.

Objetivos

- ✓ Entender los fundamentos del proceso ETL: Describir detalladamente las etapas de Extracción, Transformación y Carga, así como sus objetivos y desafíos.
- ✓ Desarrollar habilidades prácticas en la implementación de ETL: Adquirir experiencia práctica en el uso de herramientas y técnicas para construir un proceso ETL eficiente.
- ✓ Evaluar la importancia del ETL en la gestión de datos: Analizar cómo un proceso ETL bien diseñado puede mejorar la integridad y accesibilidad de los datos.
- ✓ Aplicar conocimientos teóricos a un caso práctico: Implementar un proceso ETL para una base de datos específica, documentando cada etapa del proceso y sus resultados.

Planteamiento del problema (Modelo estrella)

La base de datos de la empresa “Jardinería” actualmente es poco eficiente lo cual limita el correcto análisis de las 3 categorías de la cuales se desea obtener información generando resultados poco compresibles y apropiados para un mejoramiento sustancial de sus actividades.

Análisis del problema (Modelo estrella)

En el caso de la base de datos de “Jardinería” encontramos que el modelo usado contiene datos repetidos que ralentizan las operaciones, así como una desorganización que hace complicado obtener información que sirva para realizar un análisis de las ventas por categoría y año.

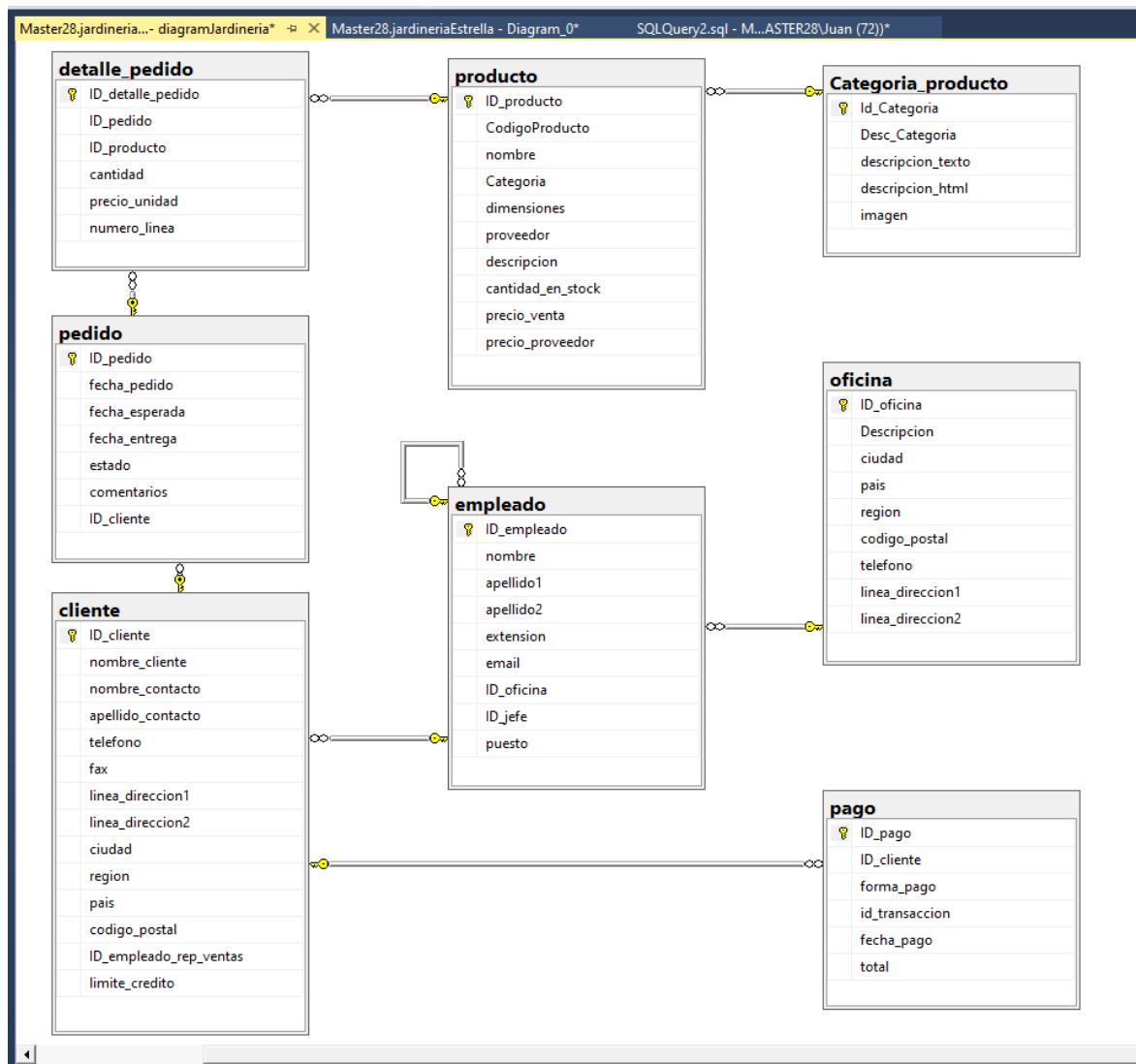


Ilustración 1 Base de datos actual jardinería

Propuesta de solución (modelo estrella)

El modelo estrella que proponemos tiene en cuenta las principales dimensiones la cual tiene una tabla de hechos llamada “ventas” con la cual no se dejan de lado información importante para la interpretación y análisis de los datos, con la tabla de hechos se almacena la actividad de las ventas de la base de datos jardinería.

Dimensiones

Empleado

- ✓ ID_Empleado (PK): INT
- ✓ Nombre: VARCHAR(50)
- ✓ Apellido: VARCHAR(50)

Cliente

- ✓ ID_Cliente (PK): INT
- ✓ Nombre_Cliente: VARCHAR(50)
- ✓ Apellido: VARCHAR(30)
- ✓ Ciudad: VARCHAR(50)
- ✓ Region: VARCHAR(50)
- ✓ Pais: VARCHAR(50)

Pedido

- ✓ ID_Pedido (PK): INT
- ✓ Fecha_Pedido: DATE
- ✓ Fecha_Esperada: DATE
- ✓ Fecha_Entrega: DATE
- ✓ ID_Cliente(FK): INT

Producto

- ✓ ID_Producto (PK): INT
- ✓ CodigoProducto: VARCHAR(15)
- ✓ Nombre: VARCHAR(70)
- ✓ Id_Categoría (FK): INT
- ✓ Proveedor: VARCHAR(50)
- ✓ Descripcion: VAR
- ✓ Cantidad: INT
- ✓ Precio_venta: NUMERIC(15,2)
- ✓ Precio_proveedor: NUMERIC(15,2)

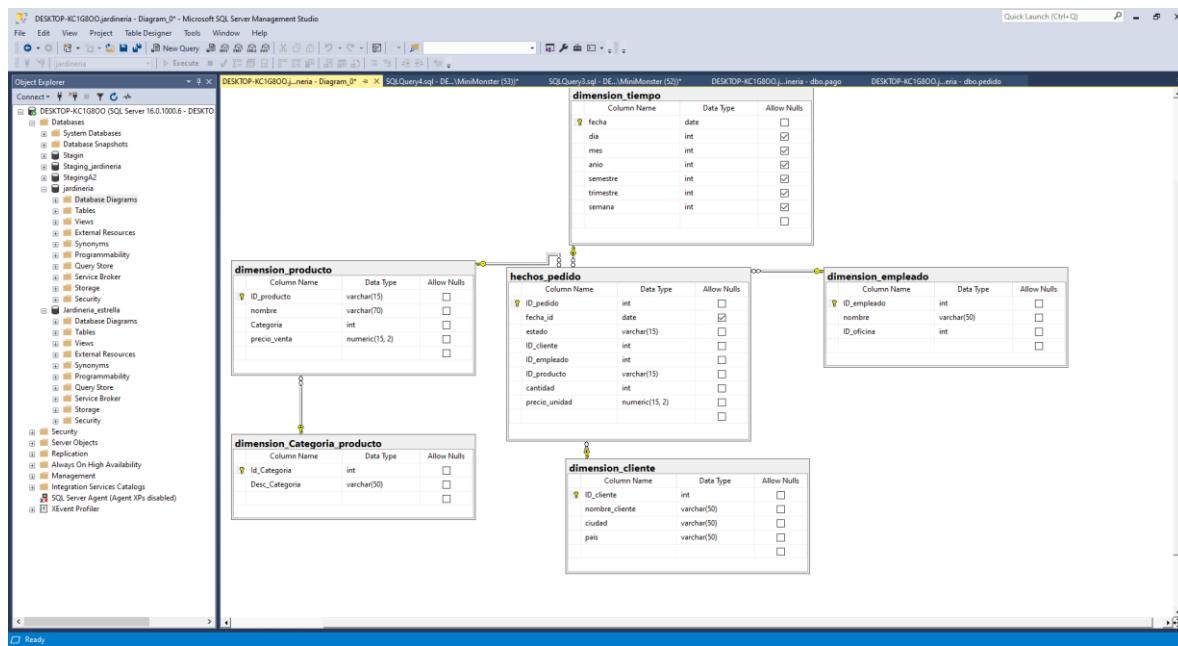
Categoría

- ✓ ID_Categoría (PK): INT
- ✓ Nombre_Categoría: VARCHAR(50)
- ✓ Descripcion: VARCHAR(50)

Tiempo

- ✓ Fecha (PK): Date
- ✓ Dia: date
- ✓ Mes: date
- ✓ Anio: date
- ✓ Semestre: date
- ✓ Trimestre: date

Diseño del modelo estrella



Query - creación modelo estrella jardinería

```
CREATE TABLE DimCliente (
    ID_cliente INTEGER PRIMARY KEY,
    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 INTEGER, limite_credito NUMERIC(15,2),
    FOREIGN KEY (ID_empleado_rep_ventas) REFERENCES empleado (ID_empleado)
);
```

```

CREATE TABLE DimProducto ( ID_producto INTEGER PRIMARY KEY, CodigoProducto VARCHAR(15),
nombre VARCHAR(70),
Categoria int,
dimensiones VARCHAR(25), proveedor VARCHAR(50), descripcion TEXT, cantidad_en_stock
SMALLINT, precio_venta NUMERIC(15,2), precio_proveedor NUMERIC(15,2),
FOREIGN KEY (Categoria) REFERENCES Categoria_producto (Id_Categoria)
);
CREATE TABLE DimTiempo ( fecha_pedido DATE, anio DATE, mes DATE, dia VARCHAR(15), semana
date,
PRIMARY KEY (fecha_pedido)
);
CREATE TABLE DimOficina (
ID_oficina INTEGER PRIMARY KEY,
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)
);
CREATE TABLE DimEmpleado ( ID_empleado INTEGER PRIMARY KEY, nombre VARCHAR(50),
apellido1 VARCHAR(50), apellido2 VARCHAR(50), extension VARCHAR(10), email VARCHAR(100),
ID_oficina INTEGER, ID_jefe INTEGER, puesto VARCHAR(50),
FOREIGN KEY (ID_oficina) REFERENCES oficina (ID_oficina), FOREIGN KEY (ID_jefe) REFERENCES
empleado (ID_empleado)
);
CREATE TABLE Ventas (
ID_pedido INTEGER PRIMARY KEY,
ID_cliente INTEGER, fecha_pedido DATE, fecha Esperada DATE, fecha_entrega DATE, estado
VARCHAR(15), comentarios TEXT, ID_producto INTEGER, ID_detalle_pedido INTEGER, ID_pago
INTEGER, ID_empleado INTEGER, ID_oficina INTEGER,
FOREIGN KEY (ID_cliente) REFERENCES DimCliente (ID_cliente),
FOREIGN KEY (fecha_pedido, fecha Esperada, fecha_entrega, estado) REFERENCES DimTiempo
(fecha_pedido, fecha Esperada, fecha_entrega, estado),
FOREIGN KEY (ID_producto) REFERENCES DimProducto (ID_producto),
FOREIGN KEY (ID_detalle_pedido) REFERENCES detalle_pedido (ID_detalle_pedido), FOREIGN KEY
(ID_pago) REFERENCES pago (ID_pago),
FOREIGN KEY (ID_empleado) REFERENCES DimEmpleado (ID_empleado), FOREIGN KEY (ID_oficina)
REFERENCES DimOficina (ID_oficina)
);

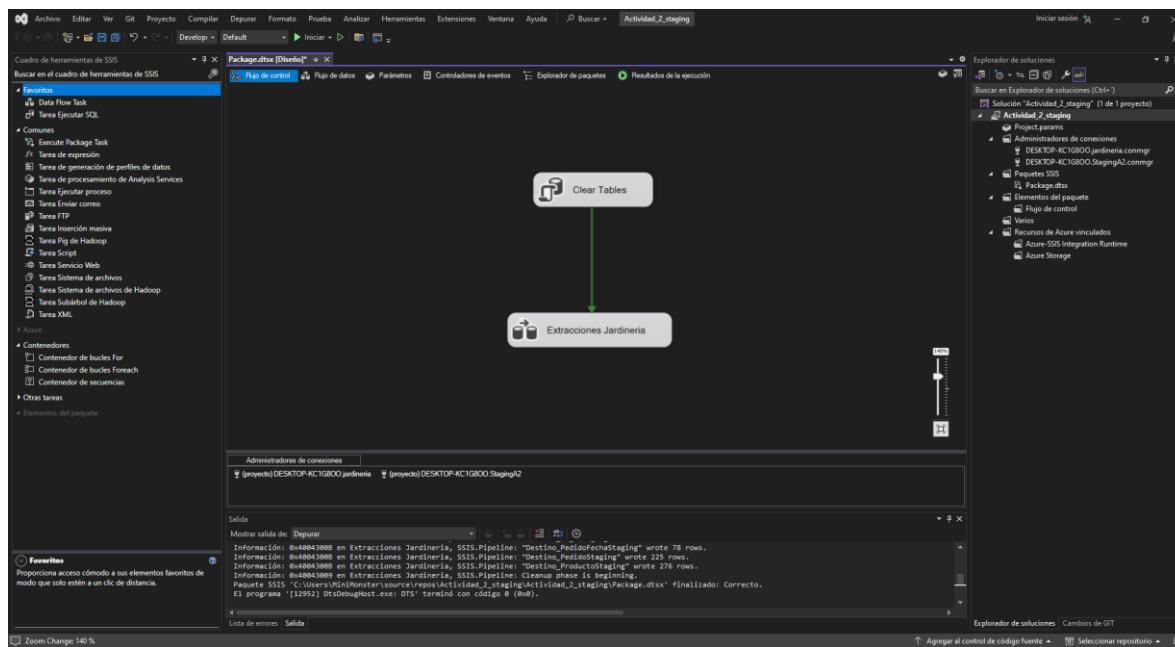
```

Propuesta de solución (Staging)

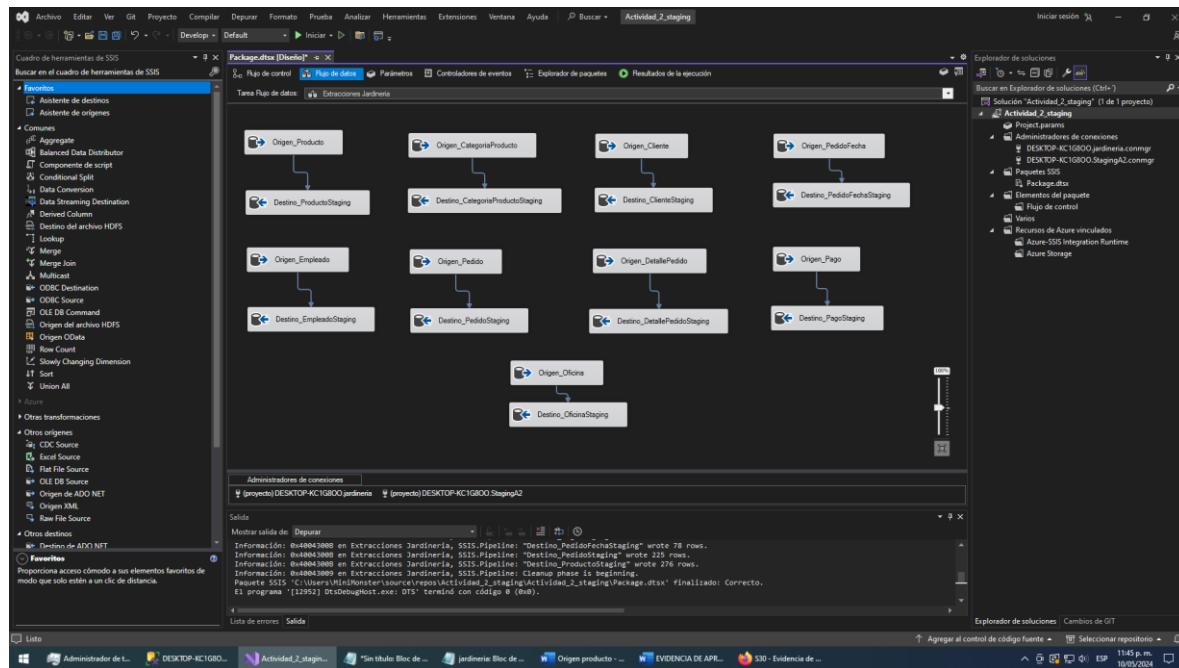
La propuesta de solución consiste en implementar un proceso de staging para reestructurar y optimizar la base de datos de Jardinería, abordando así los problemas identificados previamente. Este proceso implica la creación de tablas de destino específicas para cada conjunto de datos relevante, como productos, categorías, empleados, pedidos, clientes, detalles de pedido, fechas de pedido, pagos y oficinas. Cada tabla de destino está diseñada para almacenar datos limpios y estructurados de manera uniforme, lo que facilita su análisis y uso posterior.

Por ejemplo, se propone la creación de una tabla "Destino_ProductoStaging" para almacenar información sobre productos, con campos como ID de producto, categoría, nombre, precio de venta y precio de proveedor. De manera similar, se establecen tablas de destino para categorías de productos, empleados, pedidos, clientes, detalles de pedido, fechas de pedido, pagos y oficinas, cada una con su propio esquema de campos adecuado.

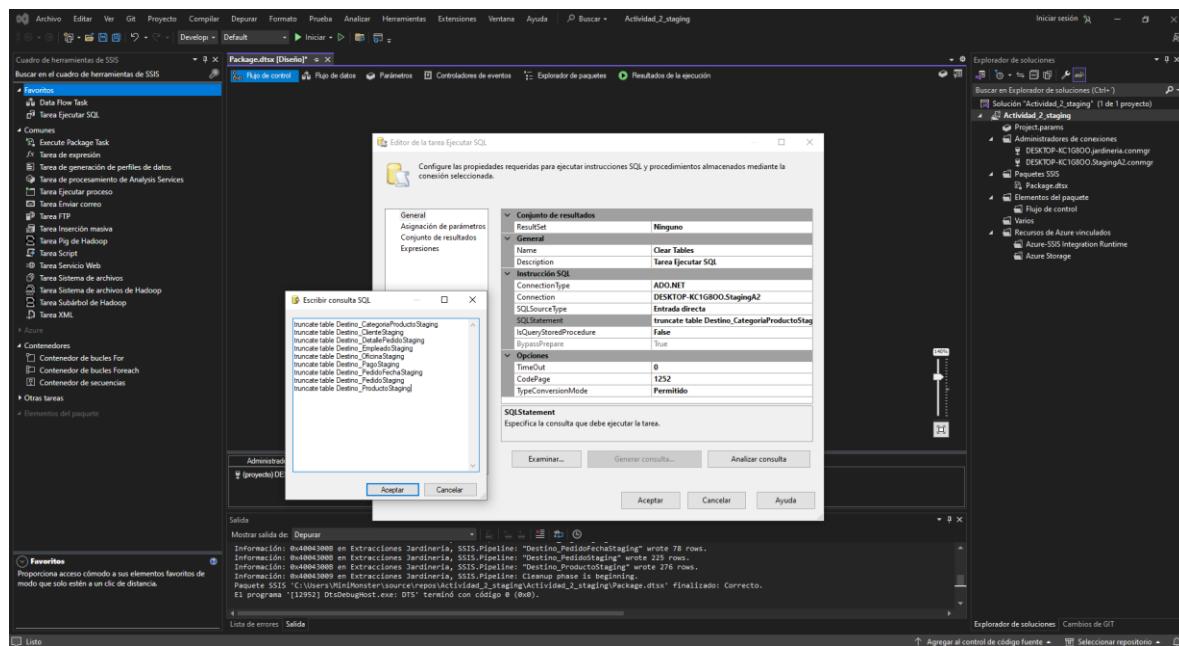
Además, se incluye un proceso de limpieza de datos mediante la eliminación de registros duplicados o innecesarios, llevado a cabo mediante la eliminación de datos de las tablas de destino utilizando la instrucción "truncate table". Esto garantiza que las tablas de destino estén vacías y listas para recibir datos limpios y actualizados durante el proceso de staging.



Staging en visual studio



Limpiar tablas: truncate



Staging en SQL Server

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the Object Explorer with the database 'DESKTOP-KC1GBOO_StagingA' selected. The right pane shows a diagram titled 'Destino..._Diagram'. The diagram contains six tables:

- Destino_ClienteStaging**: Columns: ID_cliente (int), ID_cliente_O (int), nombre_contacto (nvarchar(30)), apellido_contacto (nvarchar(30)), ciudad (nvarchar(50)).
- Destino_PedidoStaging**: Columns: ID_pedido (int), ID_pedido_O (int), fecha_pedido (date), fecha_entrega (date), estado (nvarchar(15)).
- Destino_OficinaStaging**: Columns: ID_oficina (int), ID_oficina_O (int), Descripcion (nvarchar(10)), ciudad (nvarchar(30)), pais (nvarchar(50)).
- Destino_DetallePedidoStaging**: Columns: ID_pedido (int), ID_pedido_O (int), ID_producto (nvarchar(15)), cantidad (int).
- Destino_PagoStaging**: Columns: ID_cliente (int), ID_cliente_O (int), ID_transaccion (nvarchar(50)), fecha_pago (date), total (numeric(15, 2)).
- Destino_EmppleadoStaging**: Columns: ID_empleado (int), ID_empleado_O (int), nombre (nvarchar(50)), apellido1 (nvarchar(50)), puesto (nvarchar(50)).
- Destino_PedidoFechaStaging**: Columns: ID_pedido (int), ID_fecha_pedido (int), fecha_pedido (date).

Querys para generar Staging

Origen producto

```
SELECT
p.ID_producto,
p.Categoría,
p.nombre,
p.precio_venta,
p.precio_proveedor
FROM producto p;
```

```
CREATE TABLE "Destino_ProductoStaging" (
    "ID_producto" int identity(1,1),
    "ID_producto_O" nvarchar(15),
    "Categoría" int,
    "nombre" nvarchar(70),
    "precio_venta" numeric(15,2),
    "precio_proveedor" numeric(15,2),
    primary key("ID_producto")
)
```

Categoría producto

```
SELECT
c.Id_Categoría,
c.Desc_Categoría
from Categoría_producto c ;
```

```
CREATE TABLE "Destino_CategoríaProductoStaging" (
    "Id_Categoría" int identity(1,1),
    "Id_Categoría_O" int,
    "Desc_Categoría" nvarchar(50),
    primary key("Id_Categoría")
)
```

Empleado

```
SELECT
t.ID_empleado,
t.nombre,
t.apellido1,
t.puesto
```

```
FROM empleado t;
```

```
CREATE TABLE "Destino_EmppleadoStaging" (
    "ID_empleado" int identity(1,1),
    "ID_empleado_O" int,
    "nombre" nvarchar(50),
    "apellido1" nvarchar(50),
    "puesto" nvarchar(50),
    primary key("ID_empleado")
)
```

```
Pedido
```

```
SELECT
p.ID_pedido,
p.fecha_pedido,
p.fecha_entrega,
p.estado
FROM pedido p;
```

```
CREATE TABLE "Destino_PedidoStaging" (
    "ID_pedido" int identity(1,1),
    "ID_pedido_O" int,
    "fecha_pedido" date,
    "fecha_entrega" date,
    "estado" nvarchar(15),
    primary key("ID_pedido")
)
```

```
Cliente
```

```
SELECT
c.ID_cliente,
c.nombre_contacto,
c.apellido_contacto,
c.ciudad
FROM cliente c
```

```
CREATE TABLE "Destino_ClienteStaging" (
    "ID_cliente" int identity(1,1),
    "ID_cliente_O" int,
    "nombre_contacto" nvarchar(30),
    "apellido_contacto" nvarchar(30),
    "ciudad" nvarchar(50),
```

```
    primary key("ID_cliente")
)
```

Detalle pedido

```
SELECT
d.ID_pedido,
d.ID_producto,
d.cantidad
FROM detalle_pedido d;
```

```
CREATE TABLE "Destino_DetallePedidoStaging" (
    "ID_pedido" int identity(1,1),
    "ID_pedido_O" int,
    "ID_producto" nvarchar(15),
    "cantidad" int,
    primary key("ID_pedido")
)
```

Pedido Fecha

```
SELECT
DISTINCT
f.fecha_pedido
FROM pedido f
```

```
CREATE TABLE "Destino_PedidoFechaStaging" (
    "ID_fecha_pedido" int identity(1,1),
    "fecha_pedido" date,
    primary key("ID_fecha_pedido")
)
```

Pago

```
SELECT
p.ID_cliente,
p.ID_transaccion,
p.fecha_pago
FROM pago p
```

```
CREATE TABLE "Destino_PagoStaging" (
    "ID_cliente" int identity(1,1),
```

```
"ID_cliente_O" int,  
"ID_transaccion" nvarchar(50),  
"fecha_pago" date,  
"total" numeric(15,2)  
primary key("ID_cliente")  
)
```

```
-----  
SELECT  
x.ID_oficina,  
x.Descripcion,  
x.ciudad,  
x.pais  
FROM oficina x
```

```
CREATE TABLE "Destino_OficinaStaging" (  
    "ID_oficina" int identity(1,1),  
    "ID_oficina_O" int,  
    "Descripcion" nvarchar(10),  
    "ciudad" nvarchar(30),  
    "pais" nvarchar(50),  
    primary key("ID_oficina")  
)
```

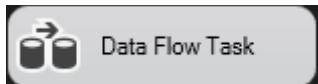
```
-----  
Clear tables
```

```
truncate table Destino_CategoríaProductoStaging  
truncate table Destino_ClienteStaging  
truncate table Destino_DetallePedidoStaging  
truncate table Destino_EmppleadoStaging  
truncate table Destino_OficinaStaging  
truncate table Destino_PagoStaging  
truncate table Destino_PedidoFechaStaging  
truncate table Destino_PedidoStaging  
truncate table Destino_ProductoStaging
```

ELEMENTOS USADOS PARA LA TRANSFORMACIÓN DE DATOS

Data Flow Task

Es el contenedor usado para las operaciones de carga y extracción de datos



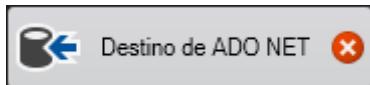
Origen de ADO NET

Con el Extraemos los datos de la base de datos de Staging



Destino De ADO NET

En destino generamos la tabla final con los datos ya transformados como resultado de las operaciones anteriores los cuales serán guardados en la BD final



Sort

Se usa para especificar por cual columna se genera el ordenamiento de los datos



Merge Join

Combina los dos flujos de datos que previamente se habían cargo desde origen ADO



Conditional Split

Lo usamos para aplicar el flujo de salida de datos con la condición de que se realice cuando exista un nuevo registro



Transformación Tabla tiempo (Transformacion_Tiempo_Staging)

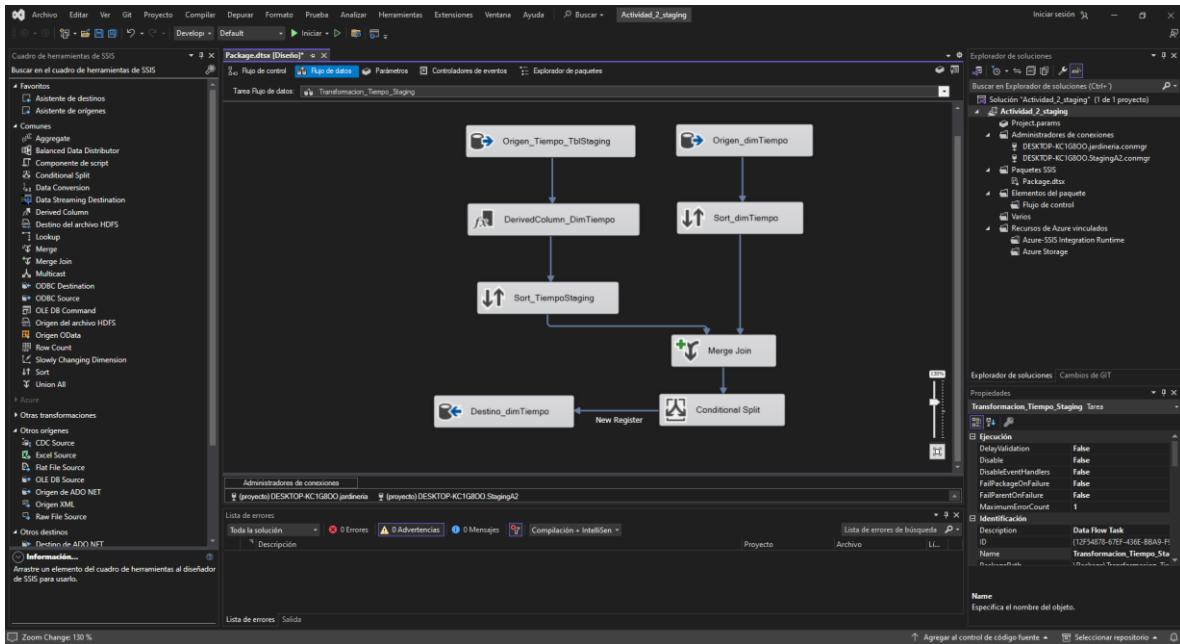
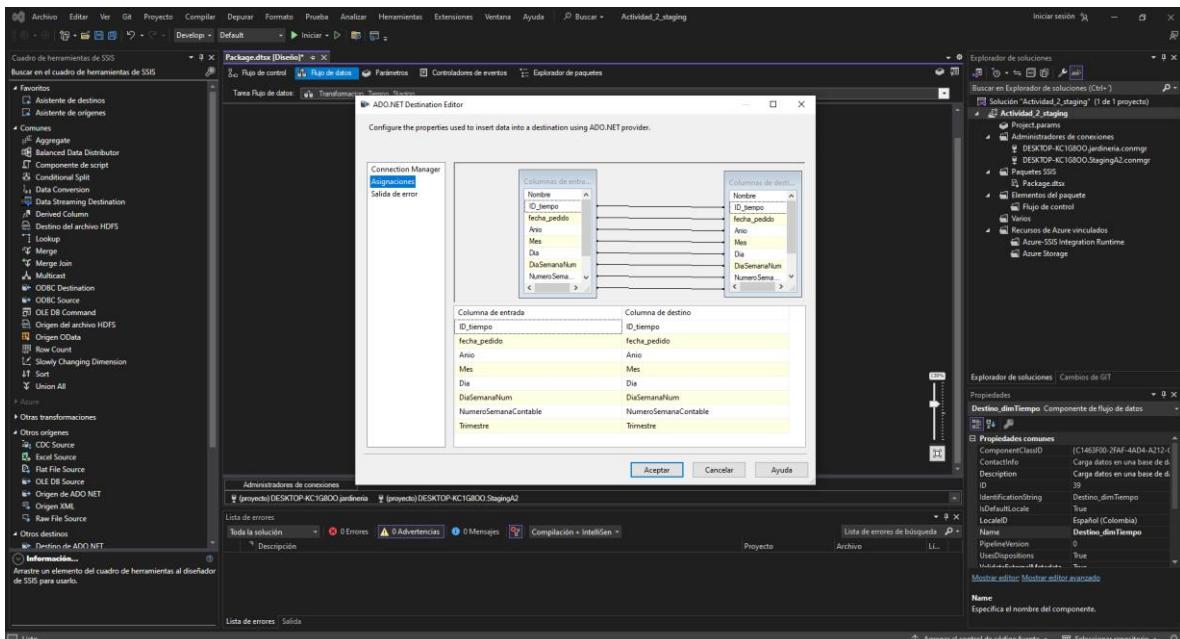


Tabla de destino tiempo (Destino_dimTiempo)



Transformación tabla cliente (Transformacion_Cliente_Staging)

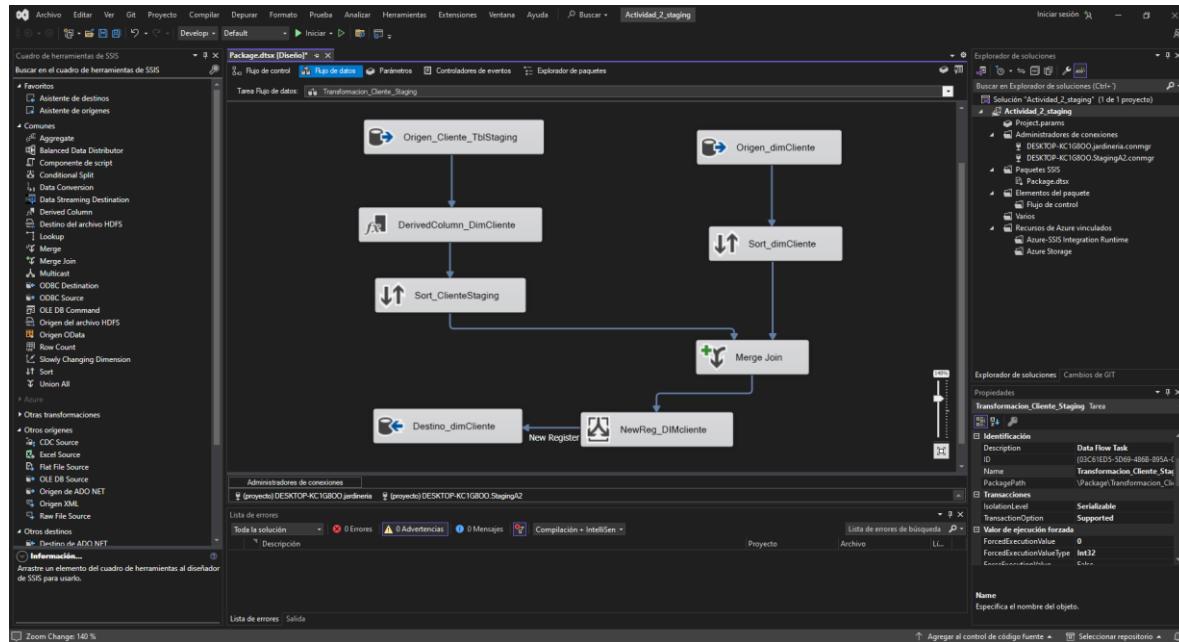
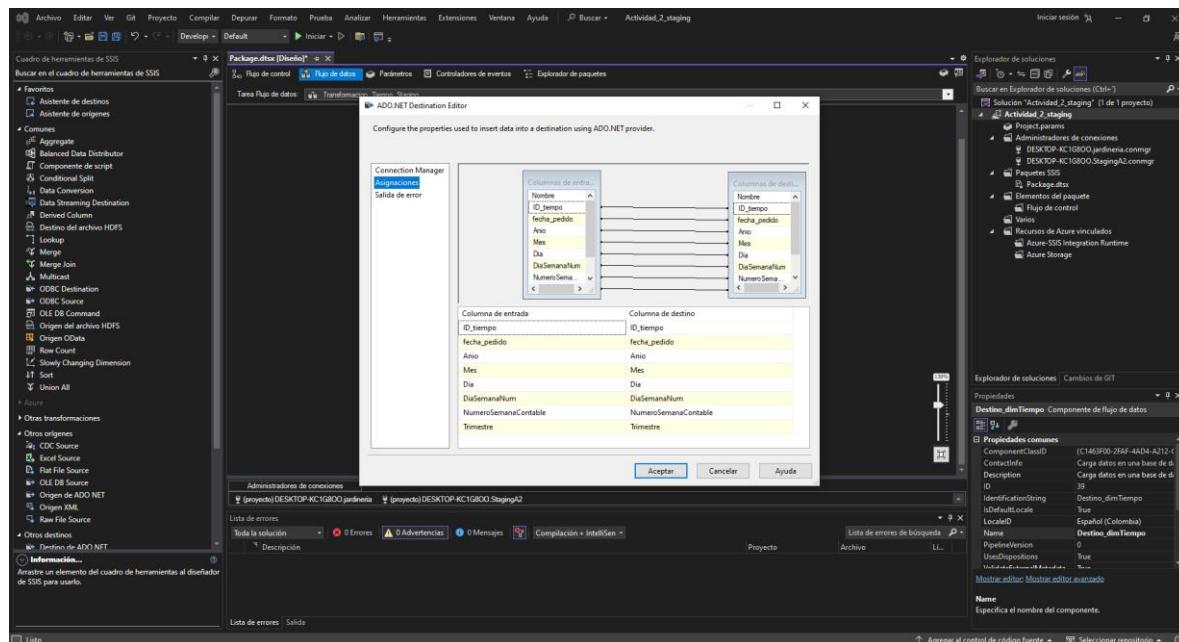


Tabla de destino cliente (Destino_dimTiempo)



Transformación tabla empleado (Transformacion_Empleado_Staging)

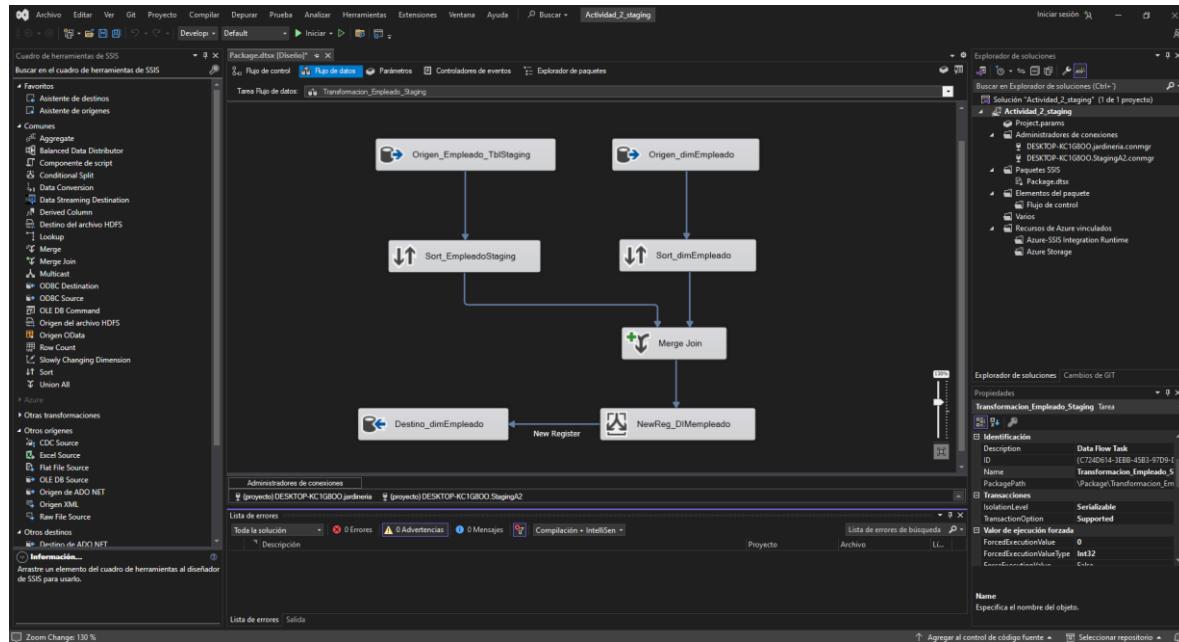
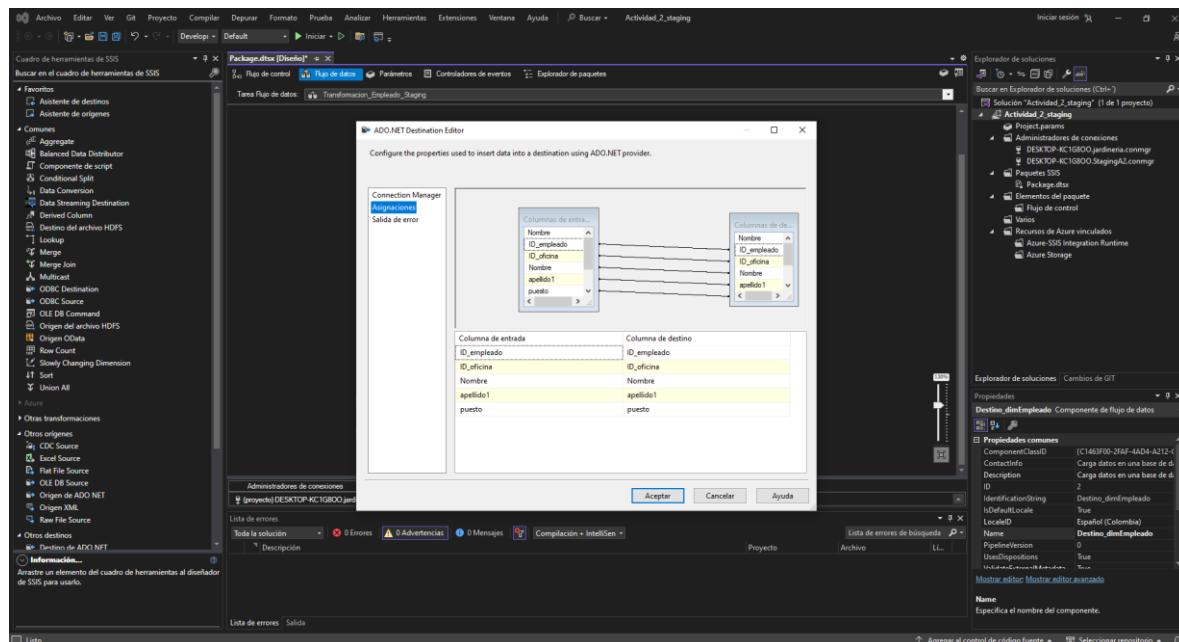


Tabla de destino empleado (Transformacion_Empleado_Staging)



Transformación tabla producto (Transformacion_Producto_Staging)

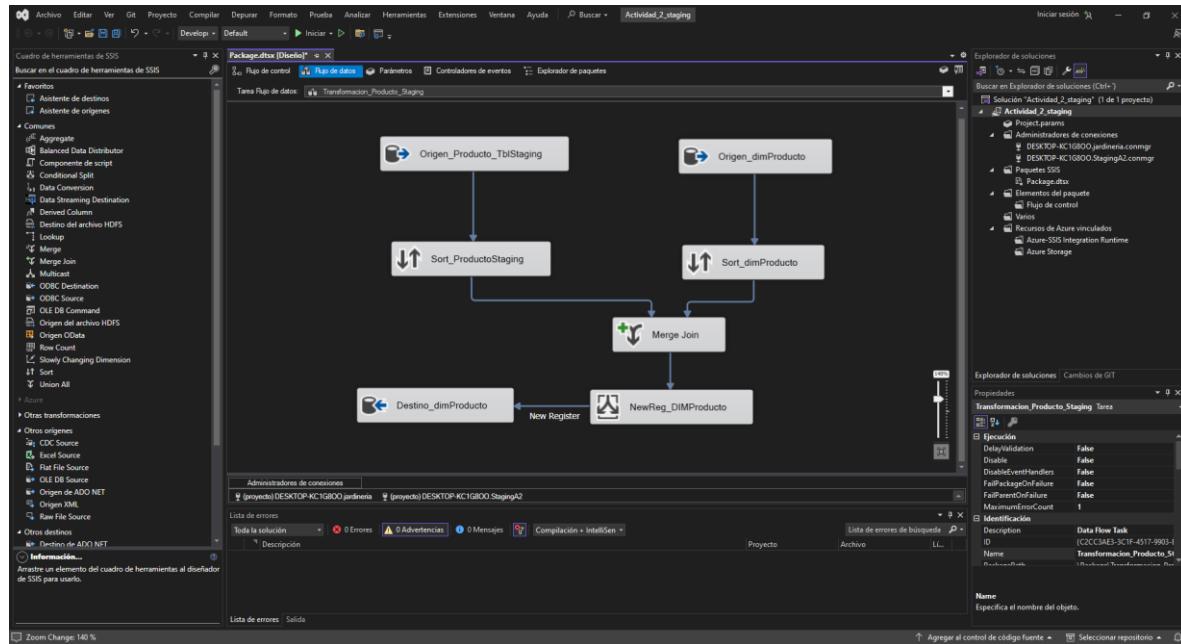
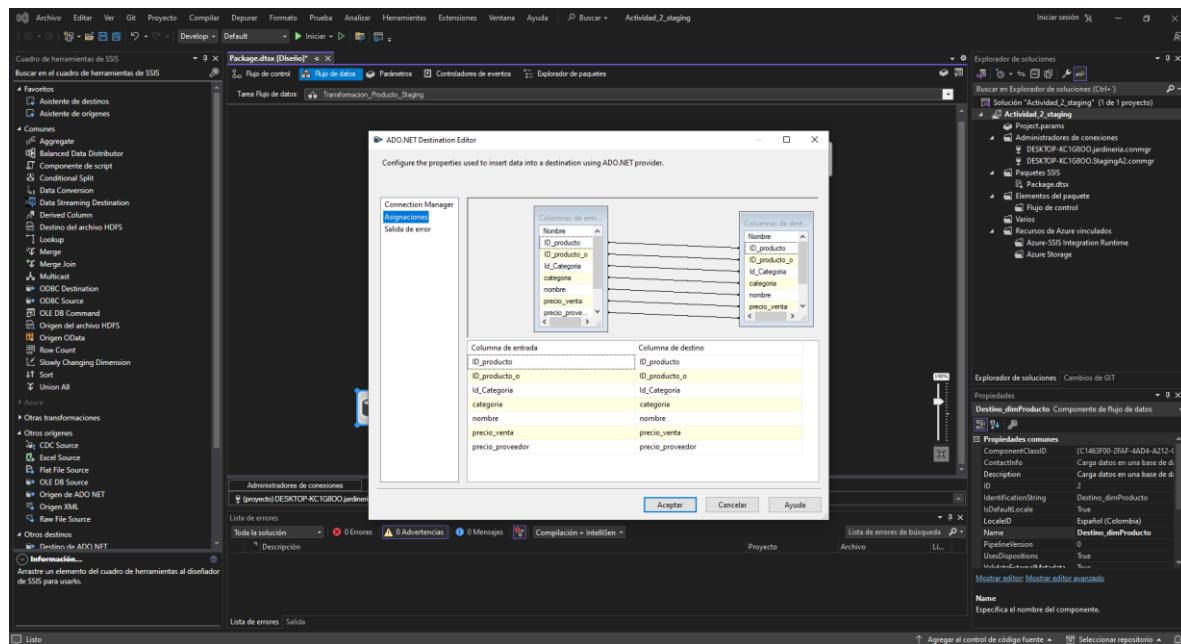


Tabla de destino producto (Transformacion_Producto_Staging)



Transformación tabla FAC (Transformacion_FACV_Staging)

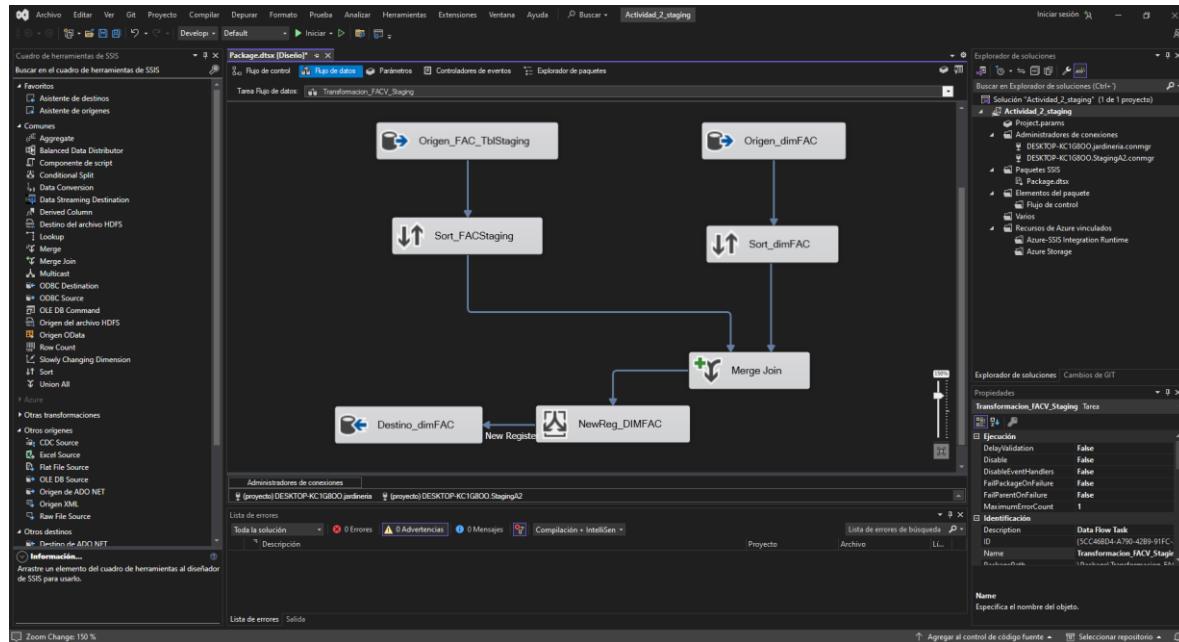
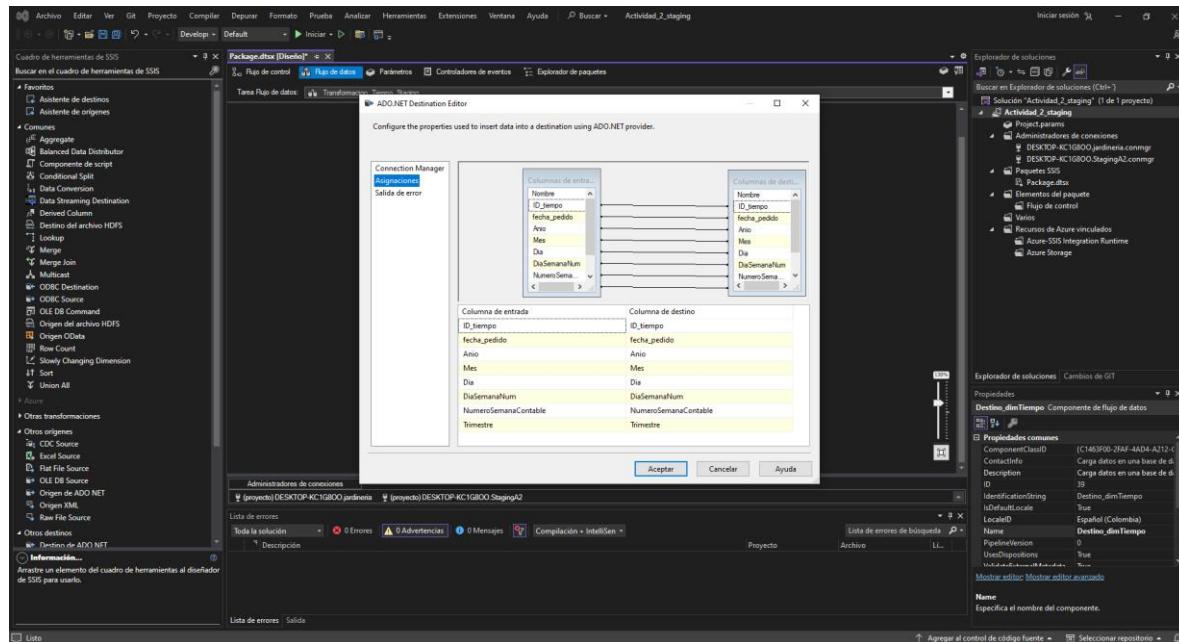


Tabla de destino FAC (Transformacion_FACV_Staging)



Querys usados para la transformación

Tabla tiempo (Transformacion_Tiempo_Staging)

```
CREATE TABLE "Destino_Tiempo" (
    "ID_tiempo" int identity(1,1),
    "fecha_pedido" date,
    primary key("ID_tiempo")
)
```

Tabla empleado (Transformacion_Empleado_Staging)

```
SELECT
    e.ID_empleado,
    o.ID_oficina,
    e.Nombre,
    e.apellido1,
    e.puesto
FROM
    Destino_EmpleadoStaging e
INNER JOIN
    Destino_OficinaStaging o
ON
    e.ID_empleado = o.ID_oficina;
```

Tabla producto (Transformacion_Producto_Staging)

```
SELECT
    p.ID_producto,
    p.ID_producto_o,
    c.Id_Categoría,
    c.Desc_Categoría AS categoria,
    p.nombre,
    p.precio_venta,
    p.precio_proveedor
FROM
    Destino_ProductoStaging p
INNER JOIN
    Destino_CategoríaProductoStaging c
ON
    p.Categoría = c.Id_Categoría_o
```

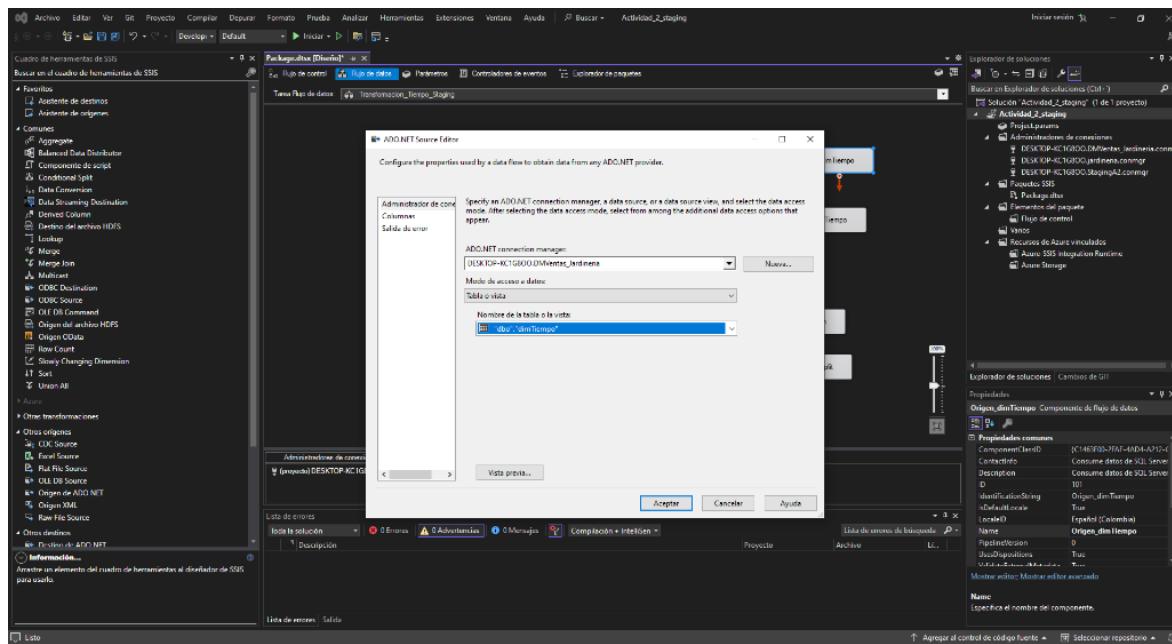
Tabla FAC (Transformacion_FACV_Staging)

```
SELECT
    p.ID_pedido,
    c.ID_cliente,
    e.ID_empleado,
    dp.ID_producto,
    dp.cantidad,
    pa.total
FROM
    Destino_PedidoStaging p
INNER JOIN
    Destino_DetallePedidoStaging dp ON p.ID_pedido = dp.ID_pedido_O
INNER JOIN
    Destino_ClienteStaging c ON p.ID_pedido_O = c.ID_cliente
INNER JOIN
    Destino_EmpleadoStaging e ON p.ID_pedido_O = e.ID_empleado
INNER JOIN
    Destino_ProductoStaging prd ON dp.ID_producto = prd.ID_producto
INNER JOIN
    Destino_PagoStaging pa ON p.ID_pedido = pa.ID_cliente_O;
```

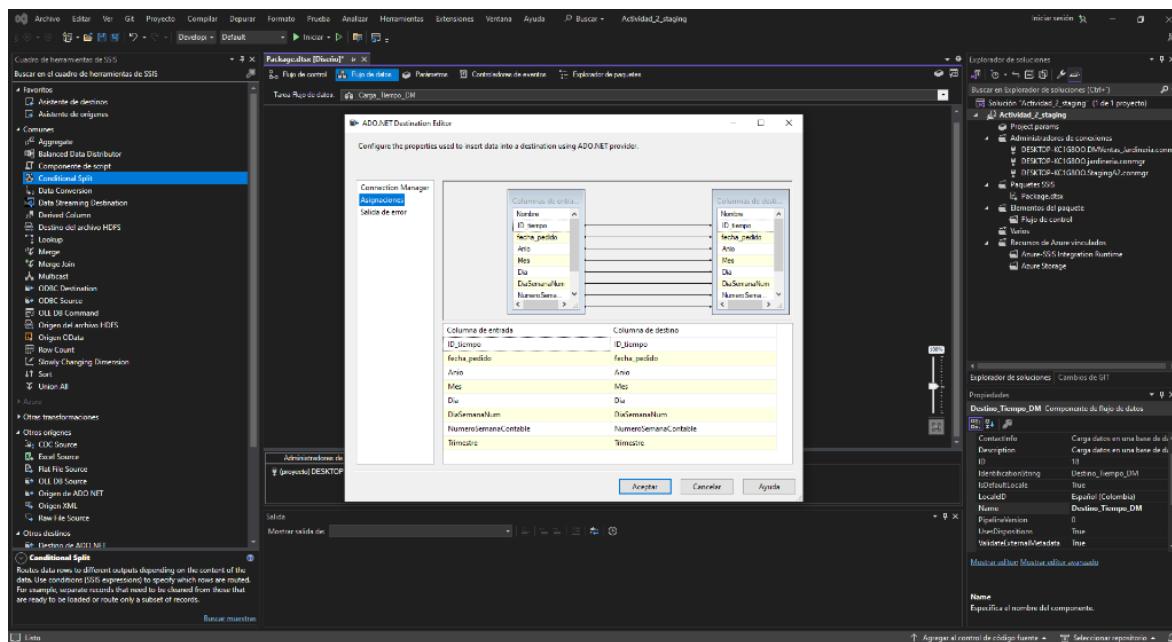
CARGA A DATAMART

Carga de la tabla Tiempo a Datamart (Carga_Tiempo_DM)

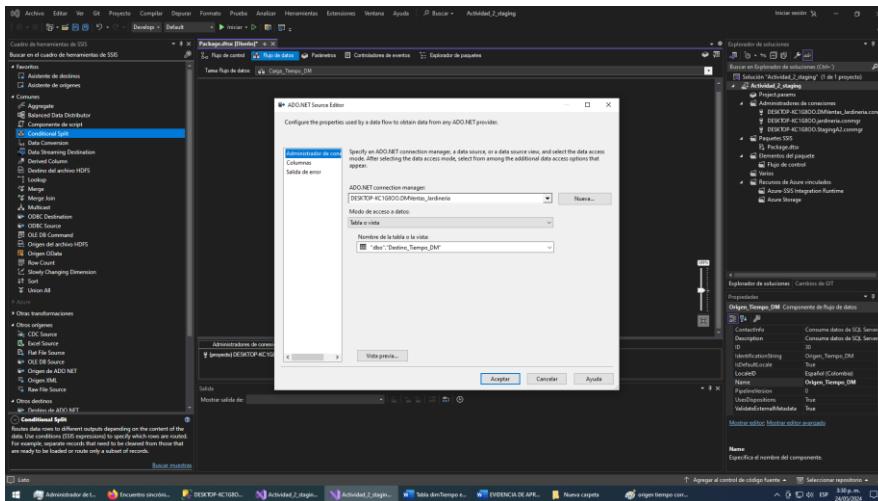
Desde origen tiempo staging



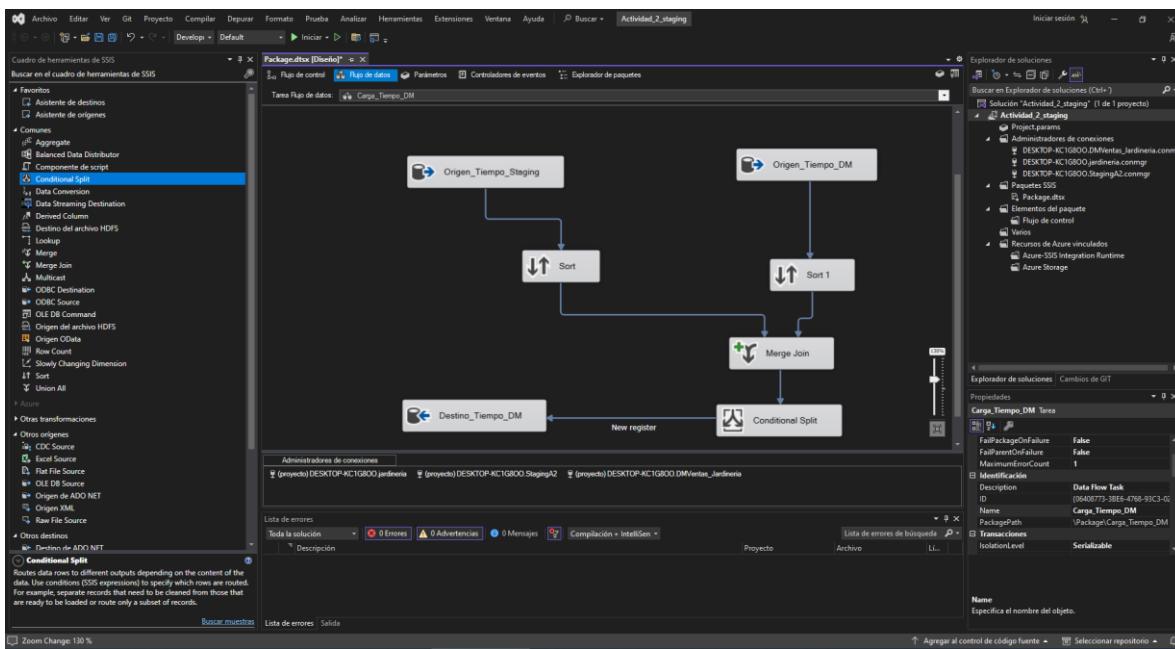
Destino tiempo Datamart (Carga_tiempo_DM)



Origen tiempo Datamart (Carga_tiempo_DM)

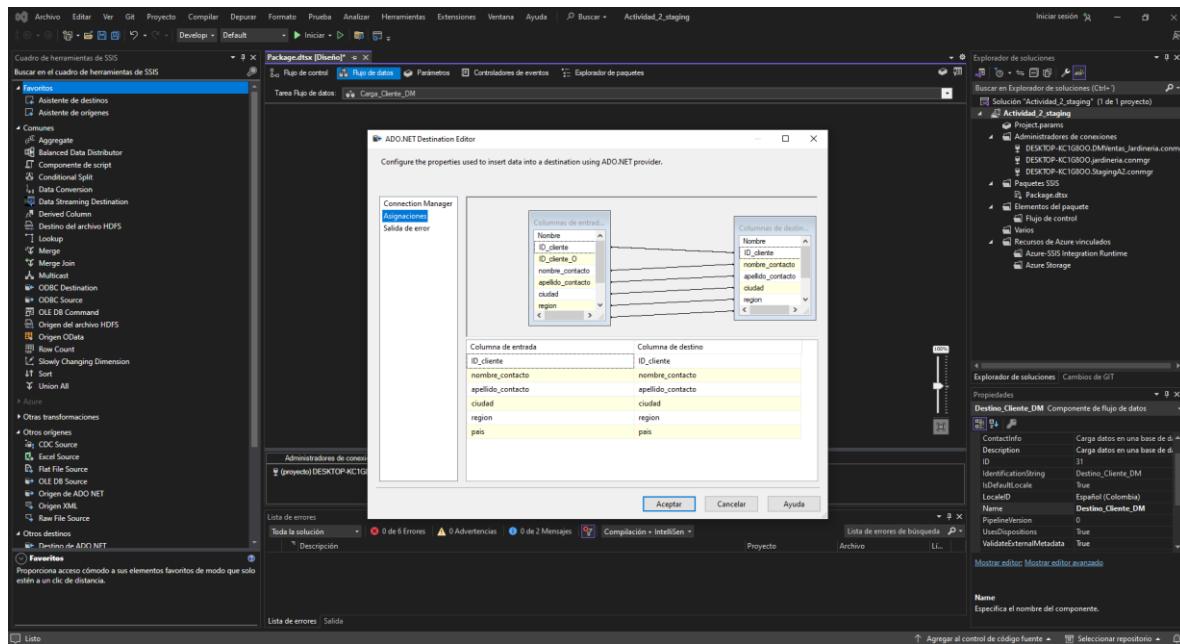


Resultado Carga Tiempo a Datamart

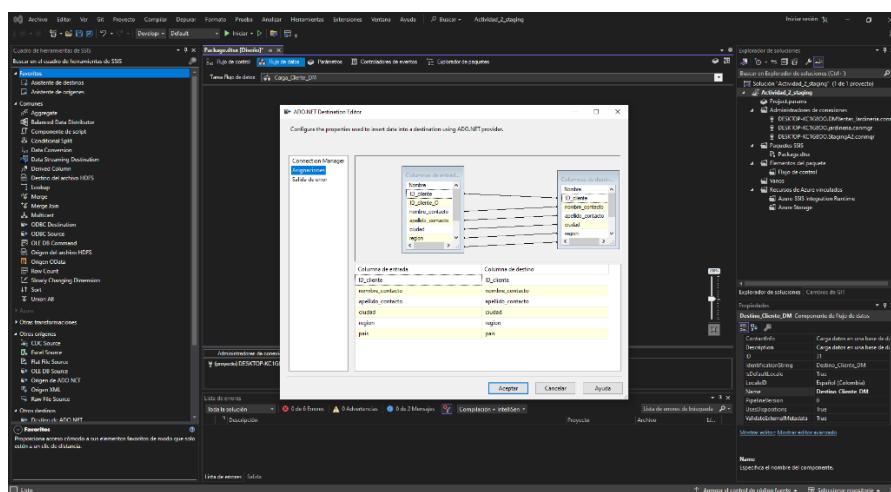


Carga de la tabla Cliente a Datamart (Carga_Cliente_DM)

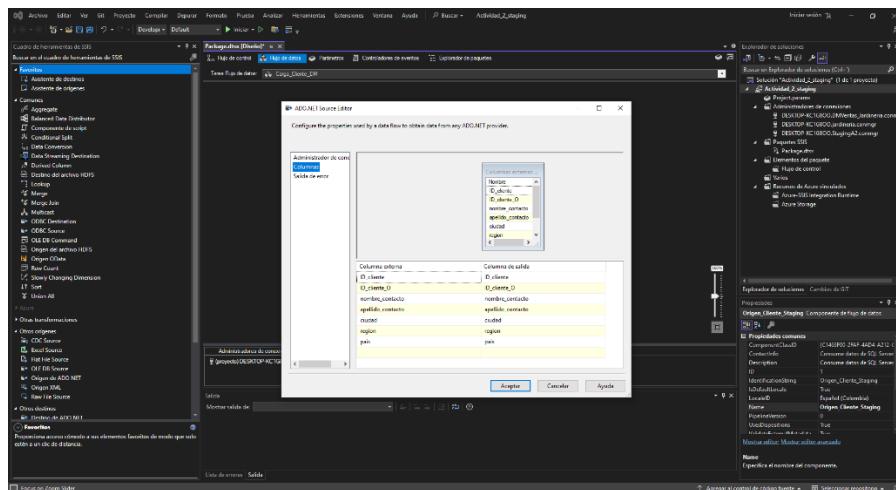
Desde origen Cliente staging



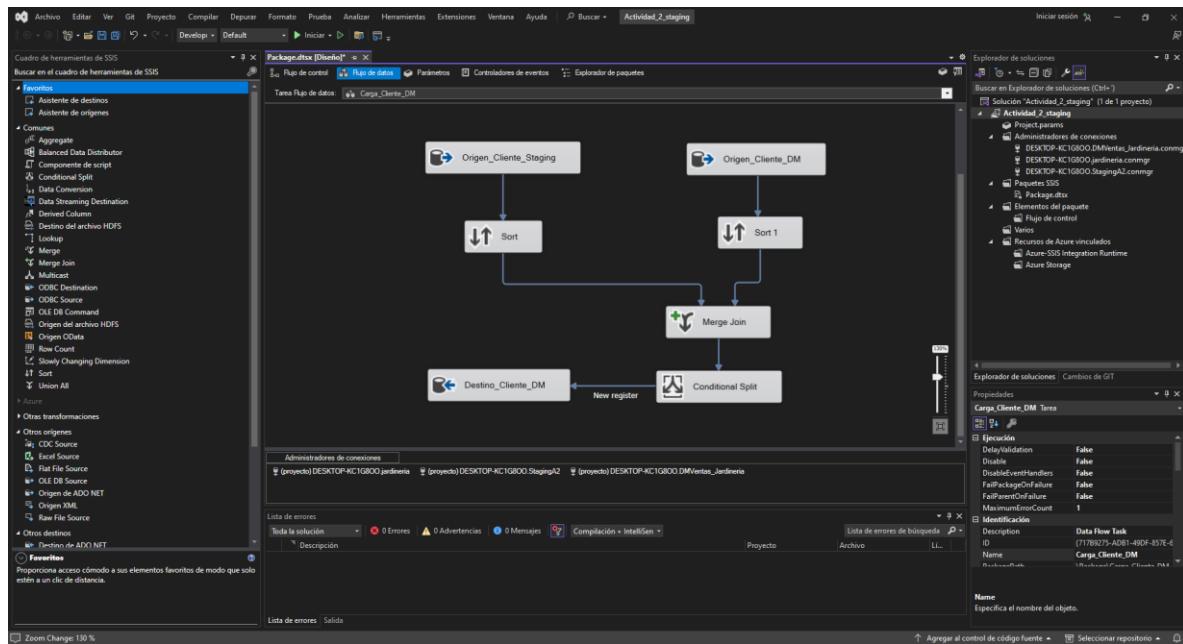
Destino Cliente Datamart (Carga_Cliente_DM)



Origen Cliente Datamart (Carga_Cliente_DM)

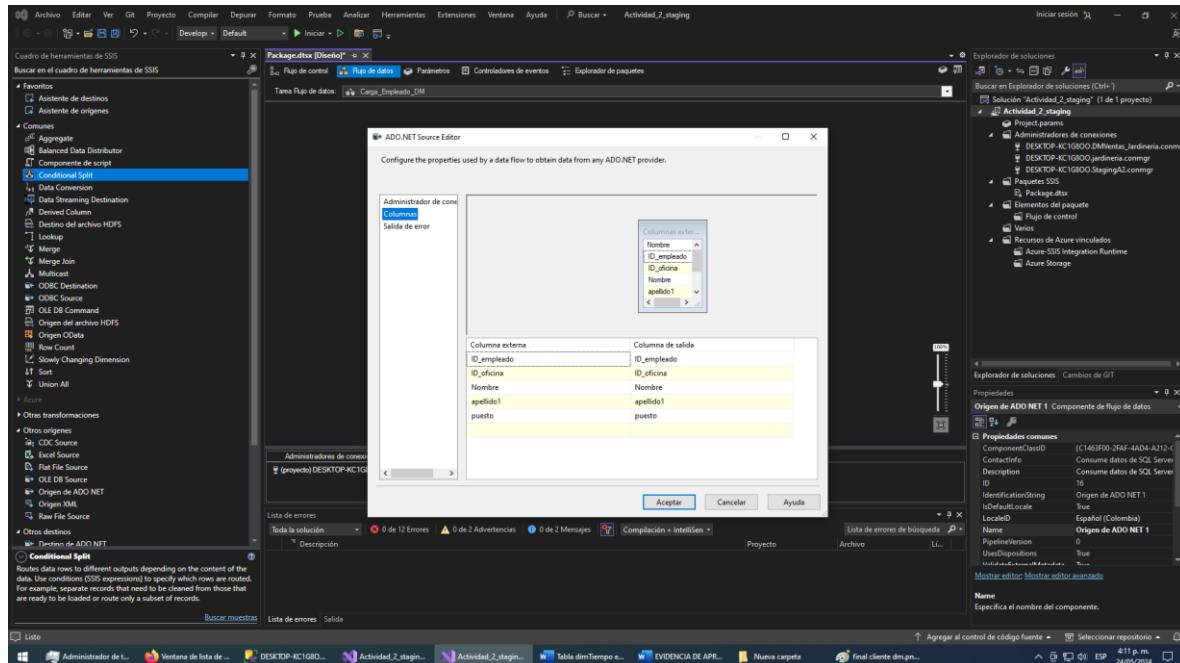


Resultado Carga Cliente a Datamart

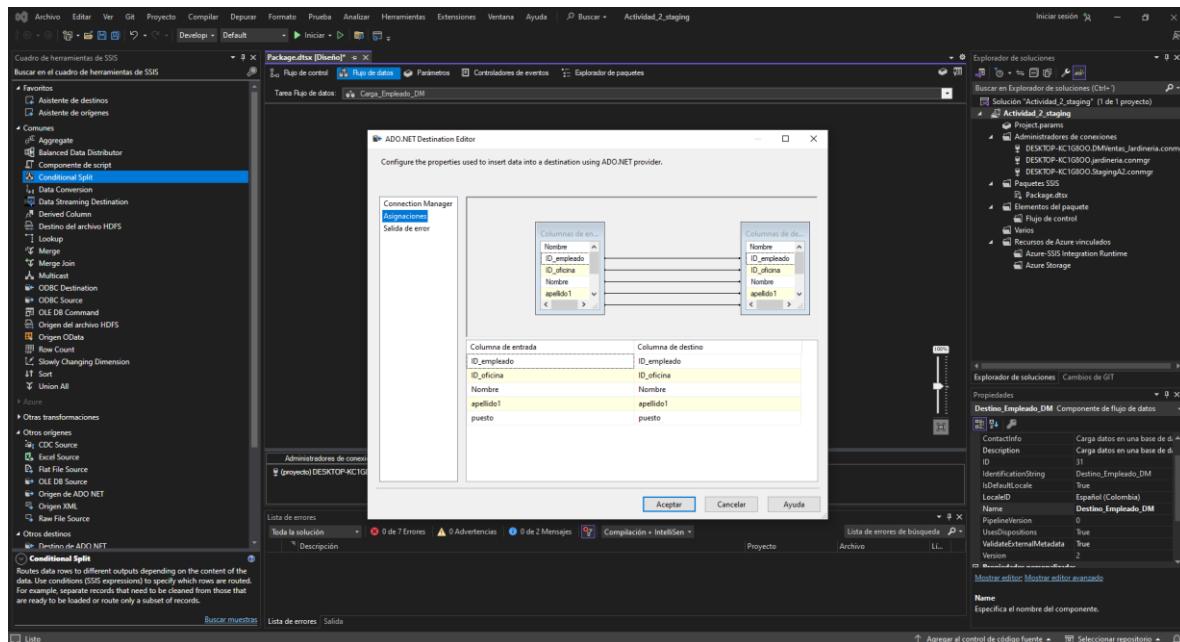


Carga de la tabla Empleado a Datamart (Carga_Empleado_DM)

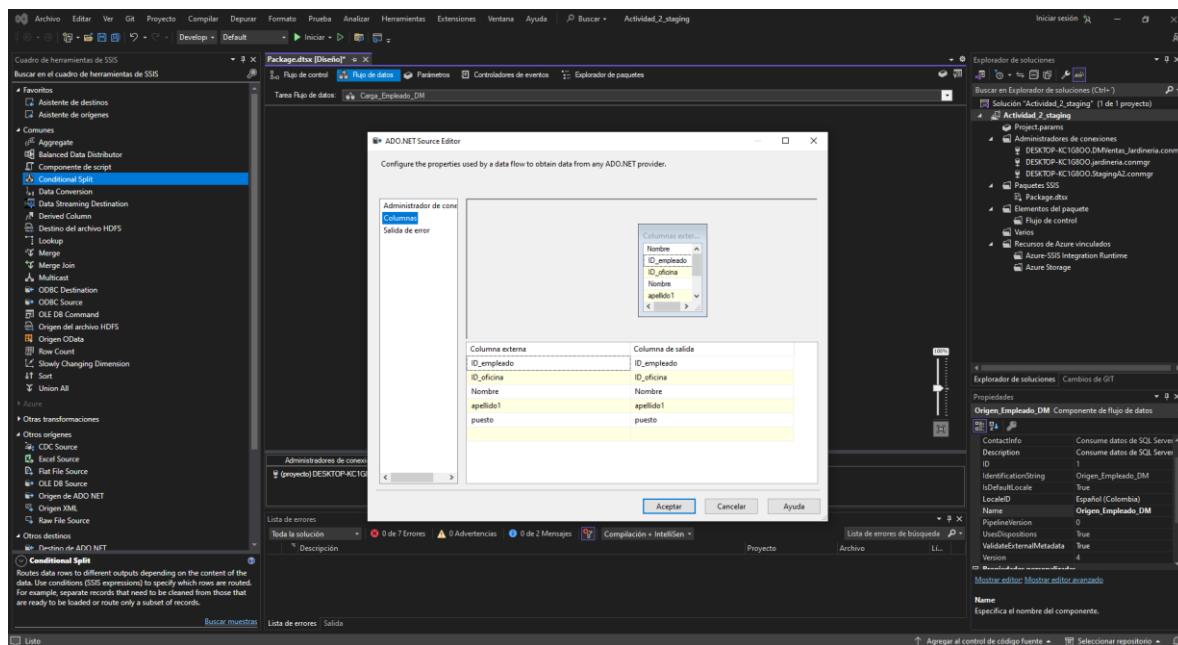
Desde origen Empleado staging



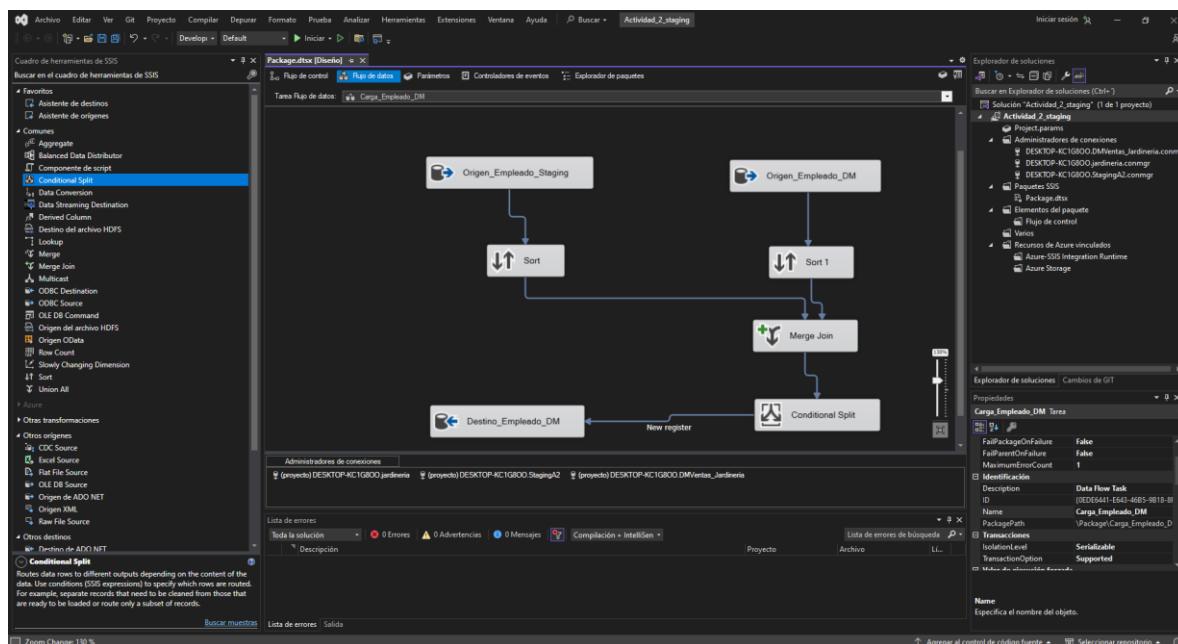
Destino Empleado Datamart (Carga_Empleado_DM)



Origen Empleado Datamart (Carga_Emppleado_DM)

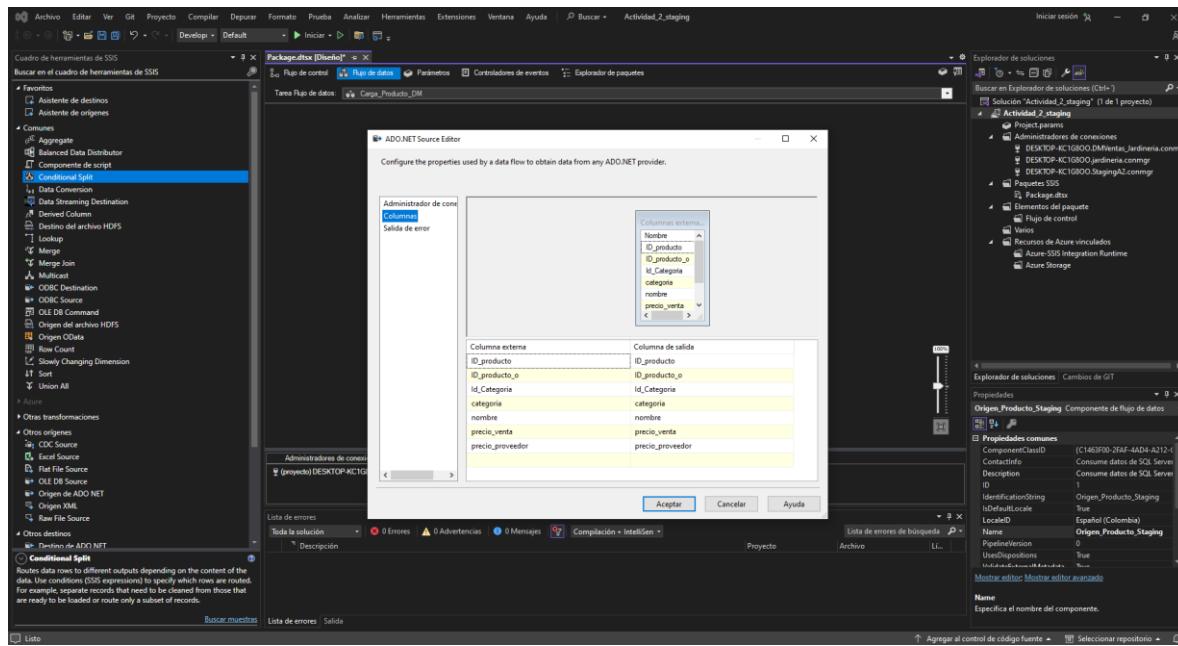


Resultado Carga Empleado a Datamart

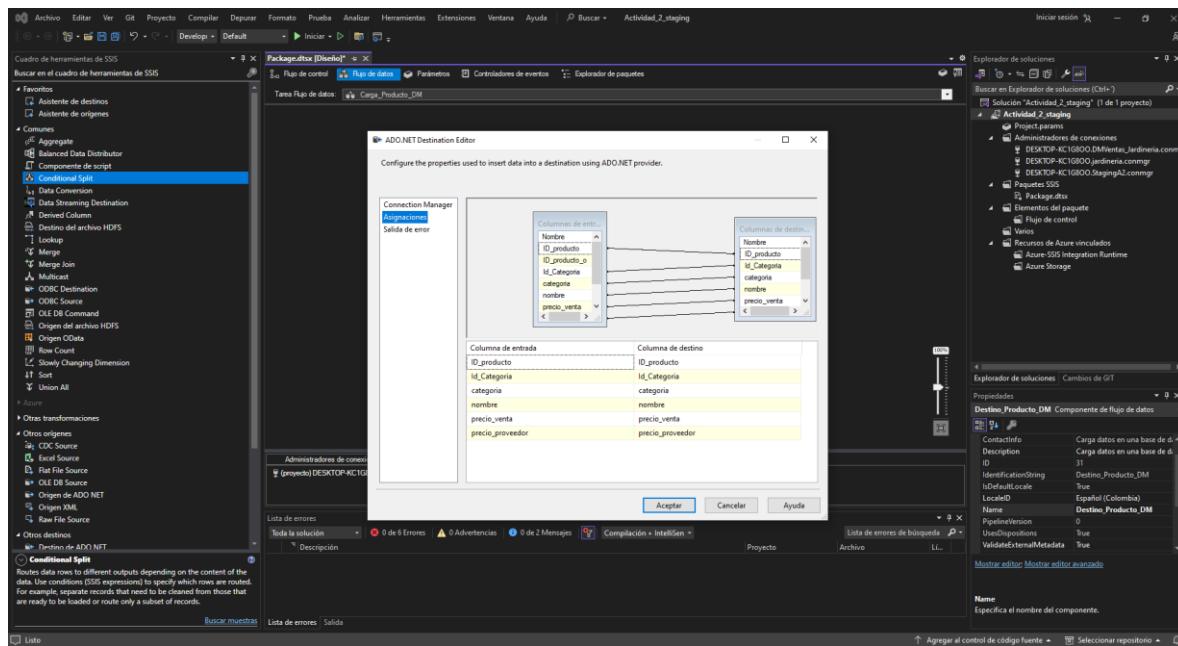


Carga de la tabla Producto a Datamart (Carga_Producto_DM)

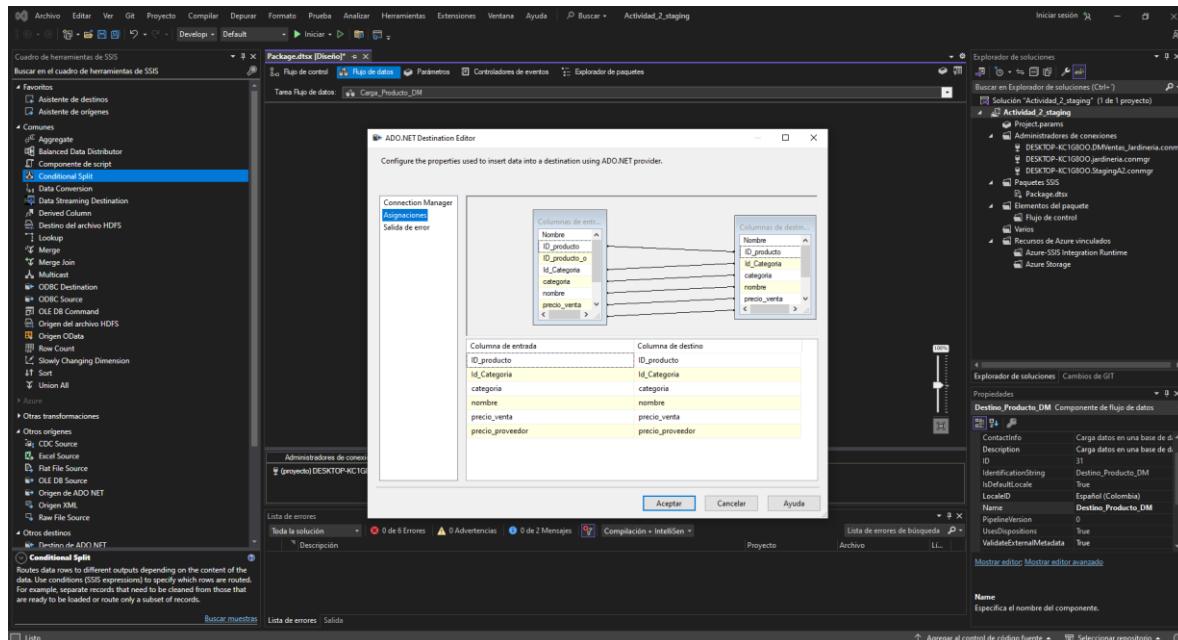
Desde origen Producto staging



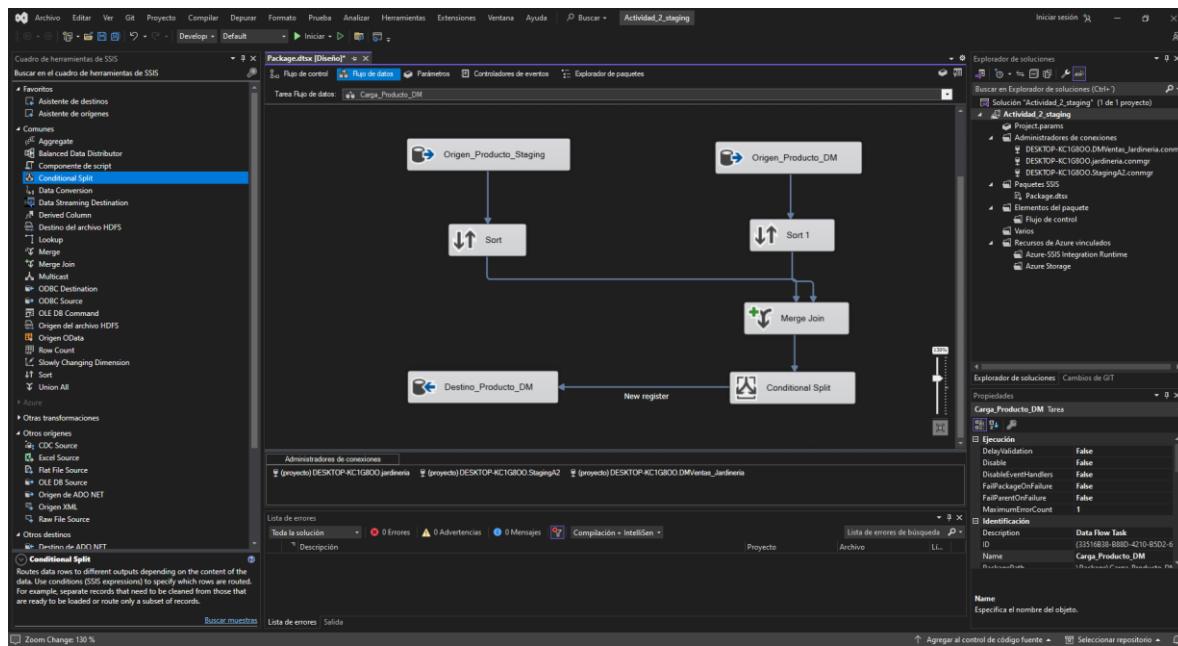
Destino Producto Datamart (Carga_Producto_DM)



Origen Producto Datamart (Carga_Producto_DM)

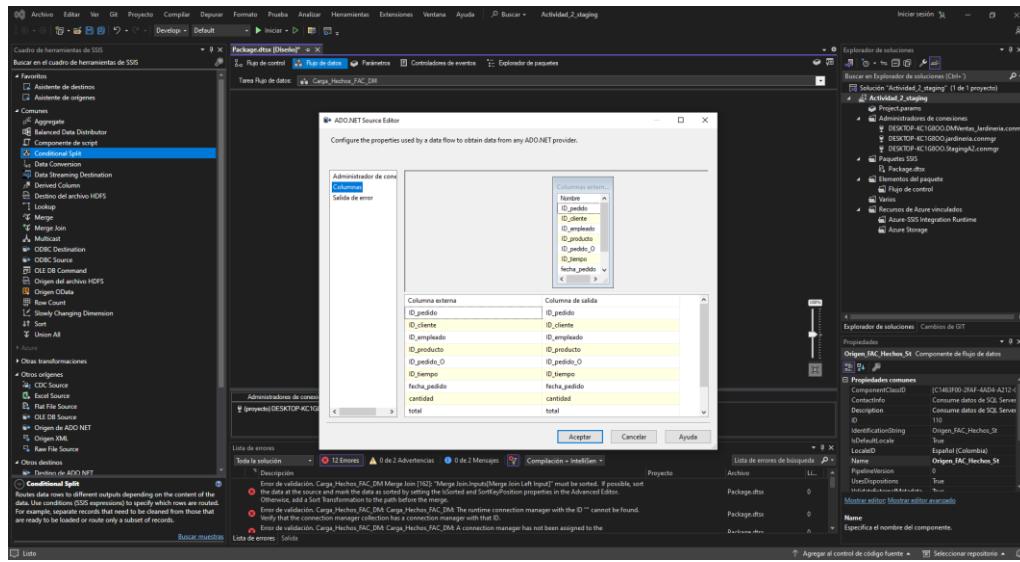


Resultado Carga Producto a Datamart

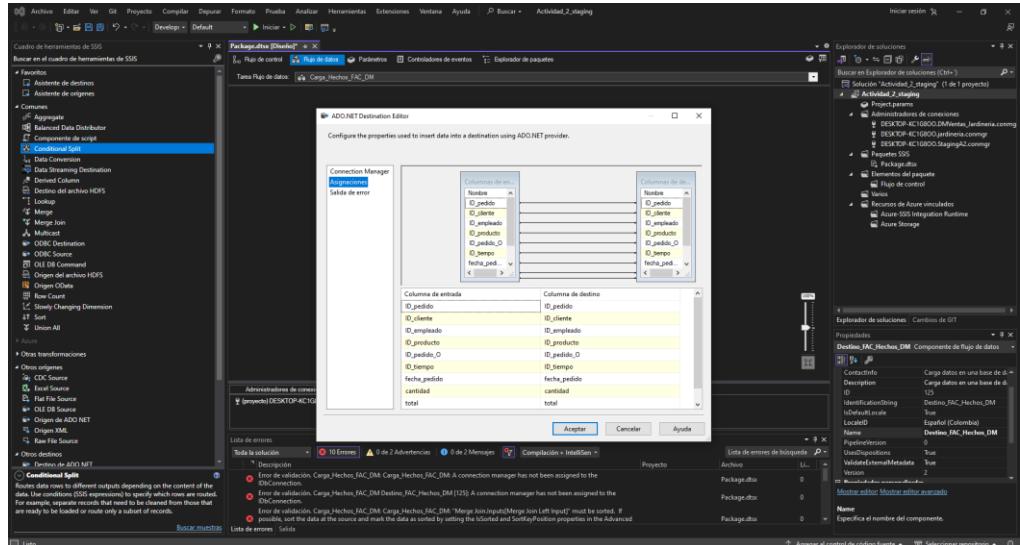


Carga de la tabla Hechos FAC a Datamart (Carga_Hechos_FAC_DM)

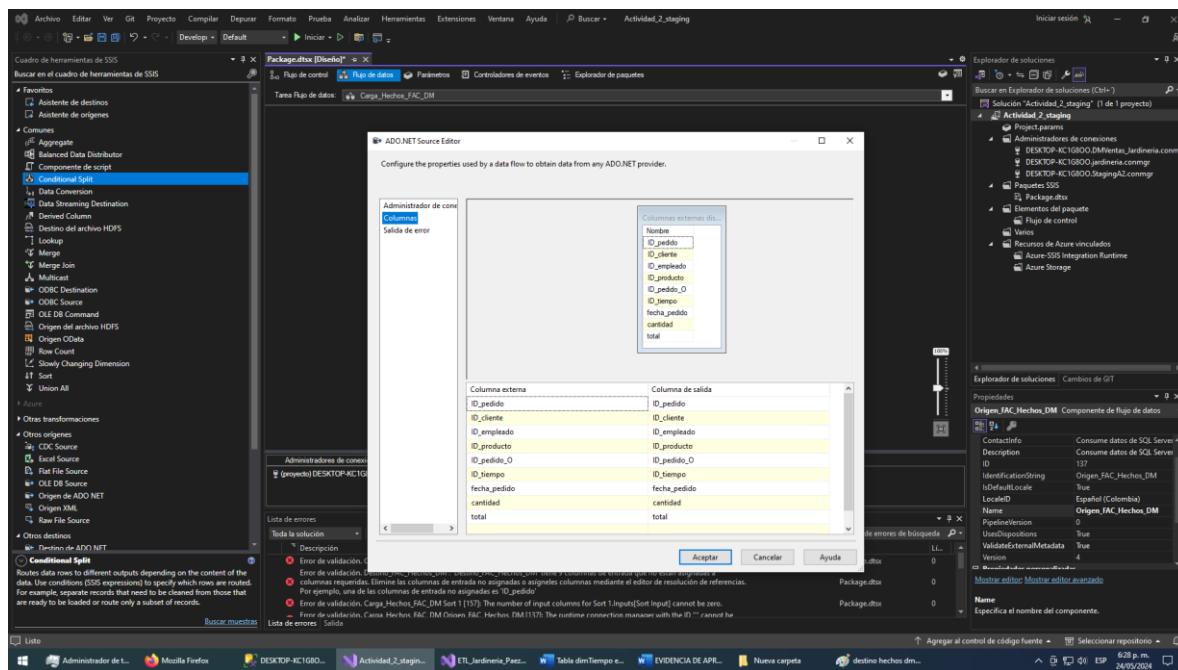
Desde origen Hechos FAC staging



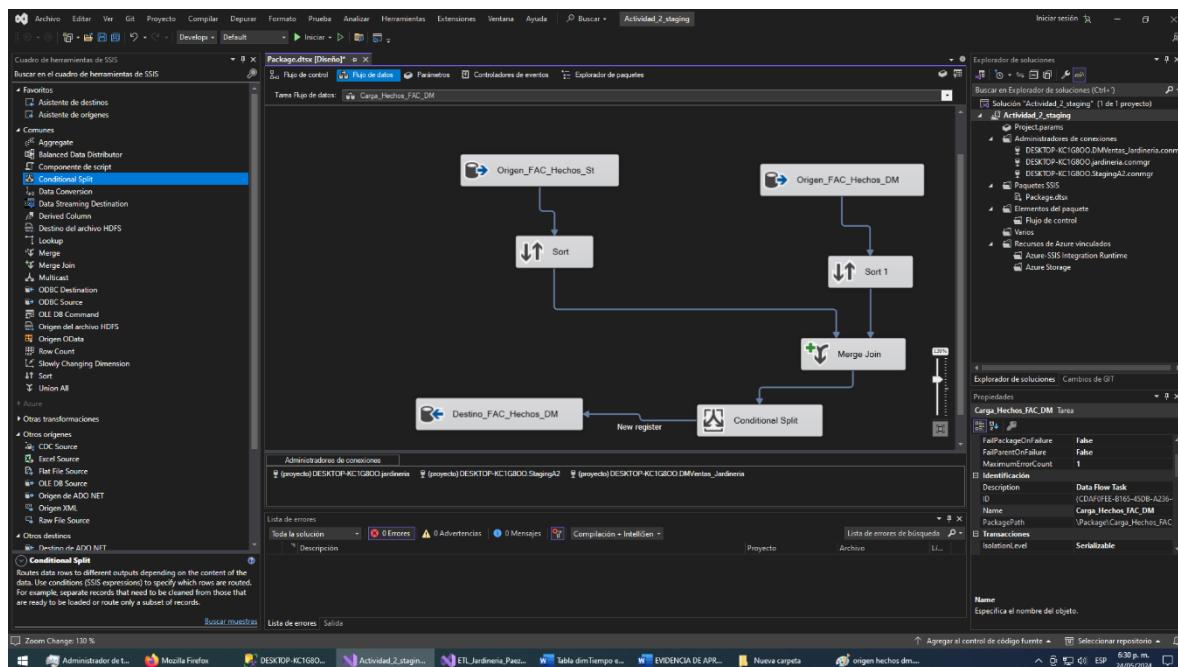
Destino Hechos FAC Datamart (Carga_Hechos_FAC_DM)



Origen Hechos FAC Datamart (Carga_Hechos_FAC_DM)



Resultado Carga Hechos FAC a Datamart



Querys usados para la carga en DM

```
CREATE TABLE "dimTiempo" (
    "ID_tiempo" int,
    "fecha_pedido" date,
    "Anio" int,
    "Mes" int,
    "Dia" int,
    "DiaSemanaNum" int,
    "NumeroSemanaContable" int,
    "Trimestre" int,
    primary key (ID_tiempo)
)
```

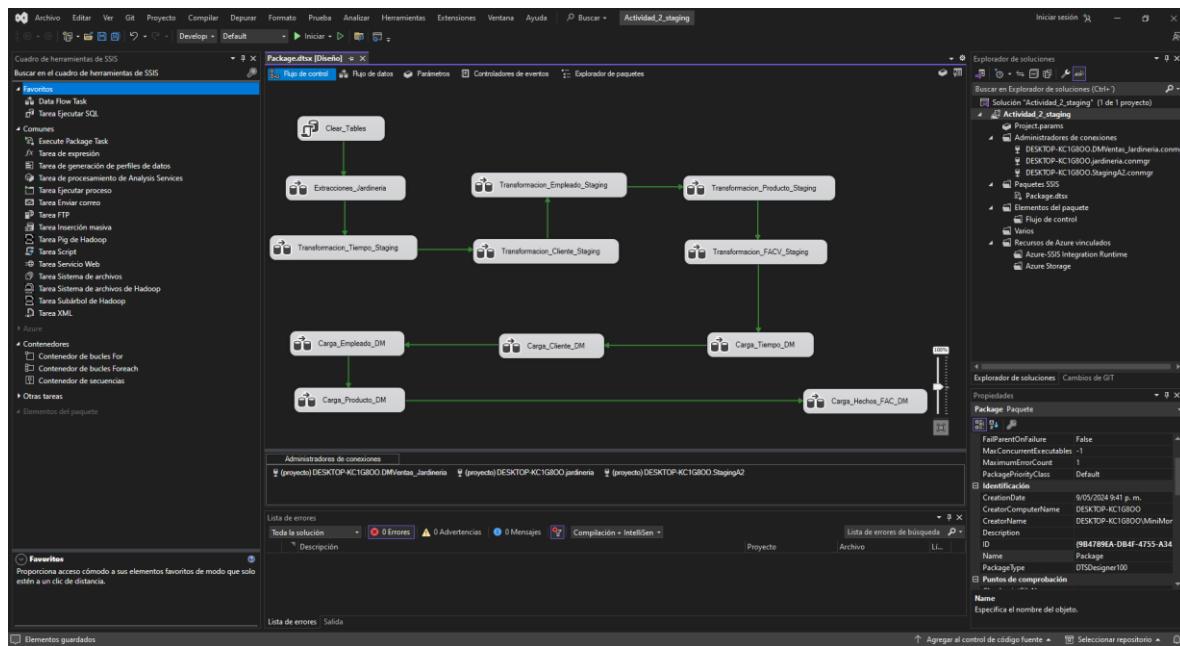
```
CREATE TABLE "Destino_Cliente_DM" (
    "ID_cliente" int,
    "nombre_contacto" nvarchar(30),
    "apellido_contacto" nvarchar(30),
    "ciudad" nvarchar(50),
    "region" nvarchar(50),
    "pais" nvarchar(50)
    primary_key("ID_cliente")
)
```

```
CREATE TABLE "Destino_Empleado_DM" (
    "ID_empleado" int,
    "ID_oficina" int,
    "Nombre" nvarchar(50),
    "apellido1" nvarchar(50),
    "puesto" nvarchar(50)
    primary key("ID_empleado")
)
```

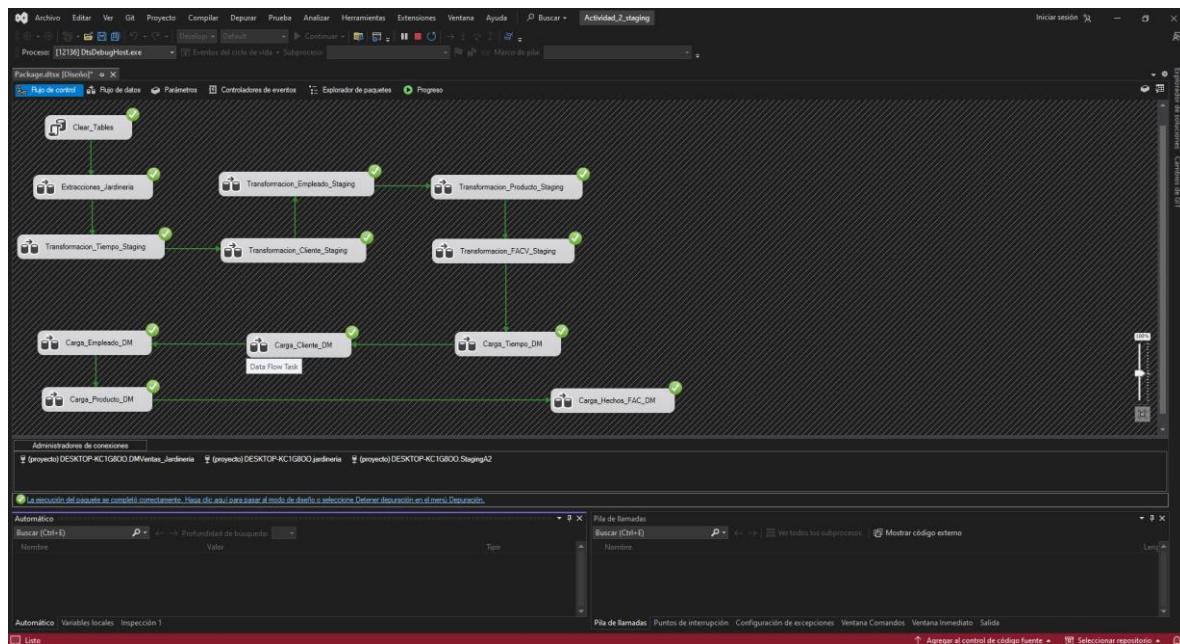
```
CREATE TABLE "Destino_Producto_DM" (
    "ID_producto" int,
    "Id_Categoría" int,
    "categoria" nvarchar(50),
    "nombre" nvarchar(70),
    "precio_venta" numeric(15,2),
    "precio_proveedor" numeric(15,2)
    primary key("ID_producto")
)
```

```
CREATE TABLE "Destino_FAC_Hechos_DM" (
    "ID_pedido" int,
    "ID_cliente" int,
    "ID_empleado" int,
    "ID_producto" int,
    "ID_pedido_O" int,
    "ID_tiempo" int,
    "fecha_pedido" date,
    "cantidad" int,
    "total" numeric(15,2),
    "ID_pedido_DM" int
primary key("ID_pedido"))
```

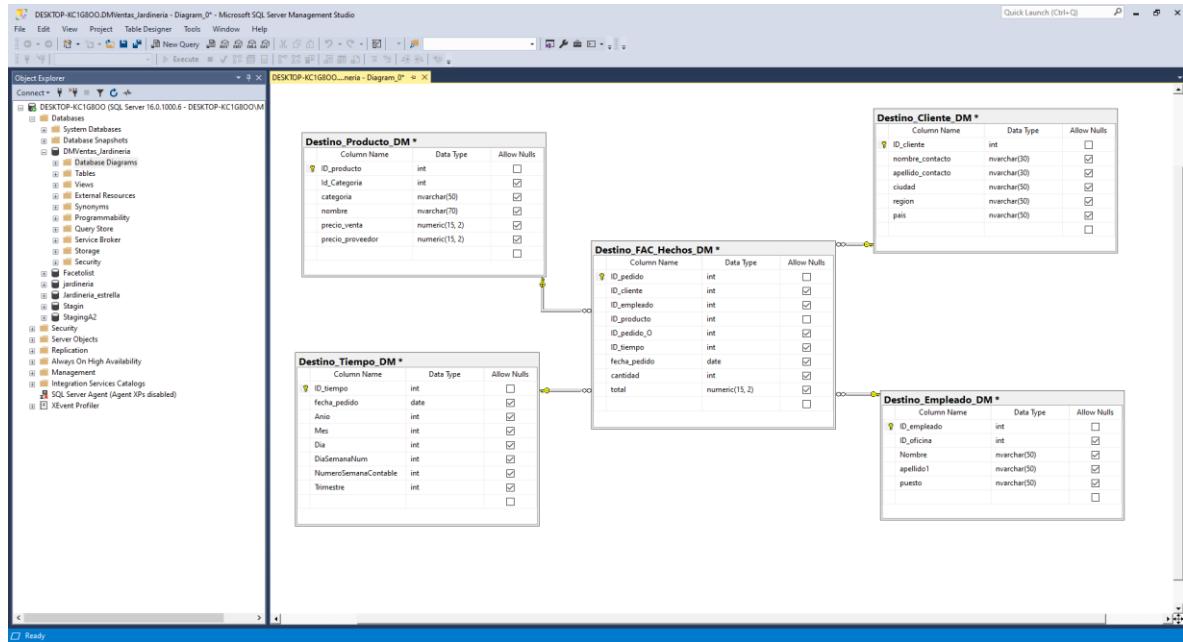
ETL Final



Ejecución Final ETL (OK)



Modelo Estrella Final BD: DMVentas_Jardineria



LINK REPOSITORIO

https://github.com/JuanCastellanos89/F20_BD2_ETL.git

Conclusiones

La implementación de un proceso ETL eficaz es esencial para manejar grandes volúmenes de datos provenientes de diversas fuentes. A través de este trabajo, se pudo observar cómo un sistema ETL bien estructurado mejora significativamente la calidad y la disponibilidad de los datos, permitiendo a las organizaciones tomar decisiones más informadas y basadas en datos confiables. Aprender estas técnicas proporciona habilidades técnicas valiosas y una comprensión profunda de la importancia de los datos en el entorno empresarial actual.

Bibliografía

Montalbán, I. L., & Rivas, J. O. (2014). BASES DE DATOS. Garceta