

BASES DE DATOS II

PREICA2501B010095

S20 - EVIDENCIA DE APRENDIZAJE 2
CREACIÓN DE UNA BASE DE DATOS DE STAGING

REALIZADO POR:

Grupo:

BASEDEDATOSIIPRE 5

JEREMY IVAN PEDRAZA HERNANDEZ

PRESENTADO A:

INSTRUCTOR

VICTOR HUGO MERCADO RAMOS

INSTITUCIÓN UNIVERSITARIA DIGITAL DE ANTIOQUIA

2025

Contenido

Planteamiento del Problema	3
Introducción	4
Objetivos	4
Análisis de la base de datos (Problema)	5
Creación de conexiones: Origen y destino	6
Creación de base de datos <i>Stagging</i>	7
Comandos SQL	7
Tabla: Categoria_producto	8
Tabla: cliente	9
Tabla: detalle_pedido	10
Tabla: Empleado	11
Tabla: Oficina	12
Tabla: Pago	13
Tabla: Pedido	14
Table: Producto	15
Código SQL para limpieza de Tablas <i>Stagging</i>	16
Optimización de tablas	17
Resultados	18
Conclusiones	24
Bibliografía	25

Planteamiento del Problema

1. Análisis de los datos almacenados en la base de datos Jardinería:
 - Revisar los datos almacenados en el Jardinería para identificar cuáles son relevantes y cuáles se deben trasladar a la base de datos *Staging*.
2. Construcción de la base de Datos *Staging*:
 - Diseñar la estructura tablas que estarán en la base de datos *Staging*.
 - Construir las consultas que permitan traer los registros de Jardinería a la Base de Datos *Staging*.
 - Ejecutar las consultas y validar que los datos queden almacenados correctamente en la Base de datos *Staging*.
 - Construir el BK de ambas bases de datos.
3. Documentación y presentación:

Redactar un documento escrito que cumpla con normas APA y contenga:

- Introducción
- Objetivos
- Planteamiento del problema
- Análisis del problema
- Propuesta de la solución con:
 - Correcciones a la entrega 1.
 - Descripción del análisis realizado a los datos *Jardinería* y cómo estos se trasladaron a la base de datos *Staging*.
 - Anexos: debe agregar los BK de las dos bases de datos y el documento del *script* de las consultas para crear la base de datos *Staging*.
 - Bibliografía.

Introducción

En este documento, se describe el proceso de extracción, transformación y carga (ETL) de los datos desde la base de datos **Jardinería** hacia la base de datos **Staging**. La base de datos Staging servirá como un entorno intermedio donde se almacenarán los datos antes de ser procesados y enviados a sistemas de análisis o almacenamiento definitivo.

Este proceso garantiza la integridad y calidad de los datos antes de su uso en reportes o análisis empresariales. Se incluyen las estructuras de las tablas en **Staging**, las consultas necesarias para trasladar los datos y los mecanismos de validación para asegurar la correcta migración

Objetivos

1. **Diseñar la estructura de las tablas en la base de datos Staging**, asegurando que reflejen la información necesaria para la integración de datos.
2. **Extraer los registros desde la base de datos Jardinería y cargarlos en Staging**, manteniendo la integridad y consistencia de la información.
3. **Validar la correcta inserción de los datos** en Staging mediante consultas de verificación.
4. **Garantizar la trazabilidad de los datos** mediante campos adicionales como fecha de carga y estado.
5. **Realizar un respaldo (backup) de ambas bases de datos** para asegurar la disponibilidad de la información en caso de errores o futuras auditorías.

Análisis de la base de datos (Problema)

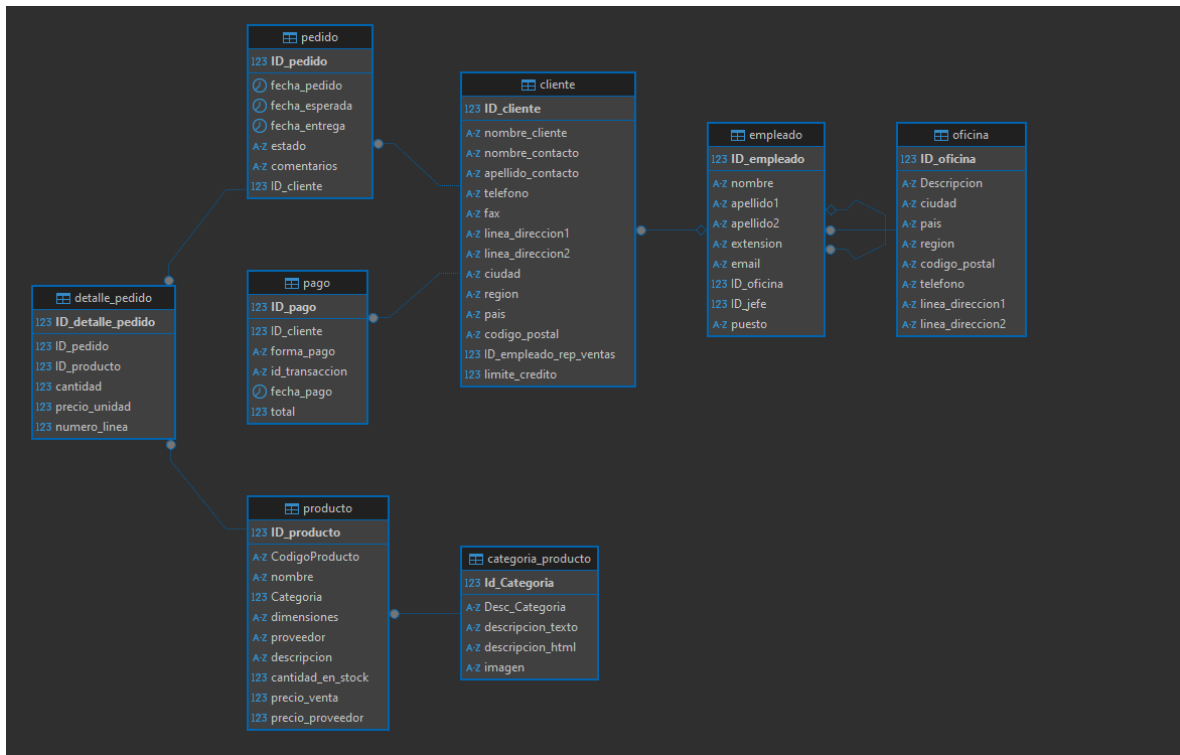


Ilustración 1 Modelo Transaccional Jardinería

1. Se ha cargado la base de datos en la instancia SQL y se insertan los datos, así mismo el código cuenta con relaciones en llaves foranes preestablecidas. El producto de esto se muestra en la presente imagen (Ilustración 1)

Creación de conexiones: Origen y destino

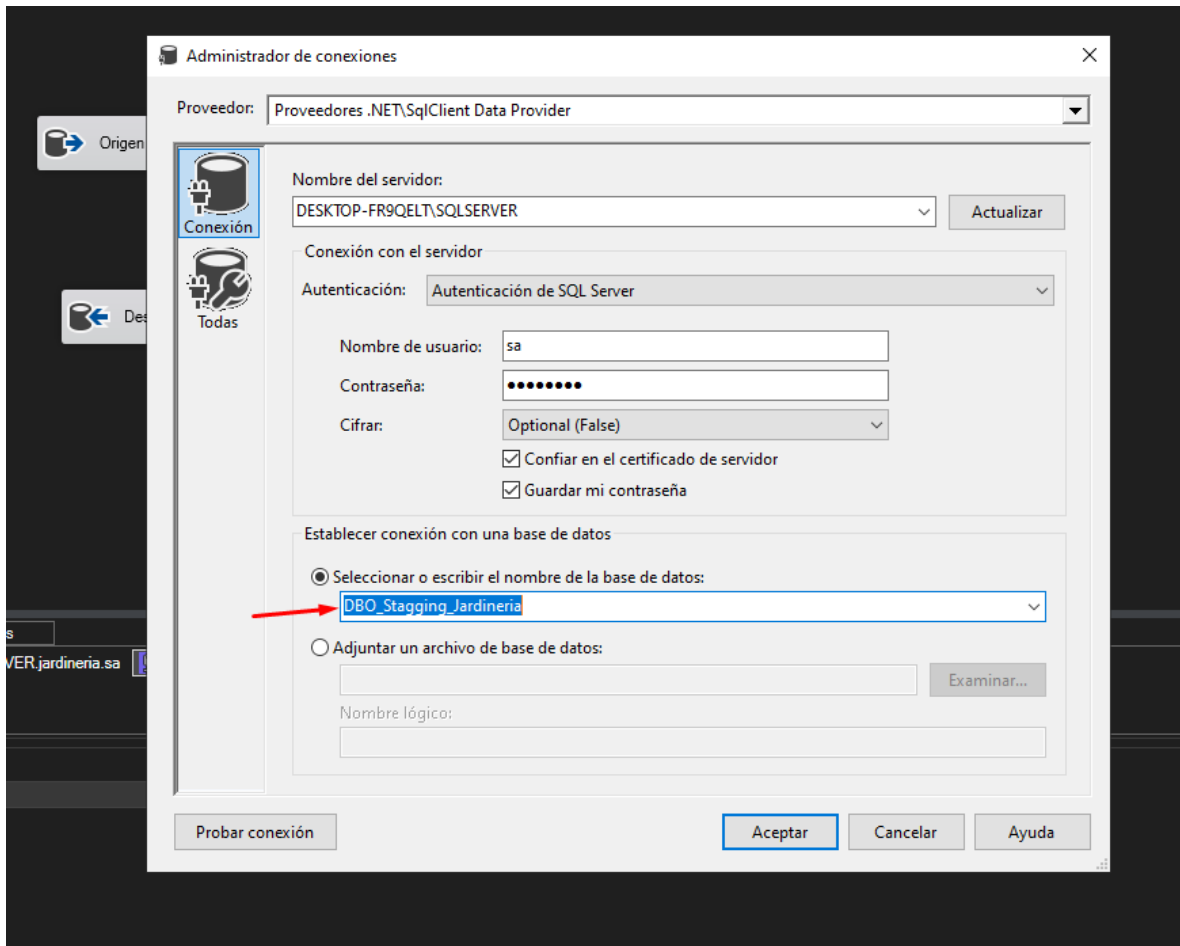


Ilustración 2 Conexión origen y staging

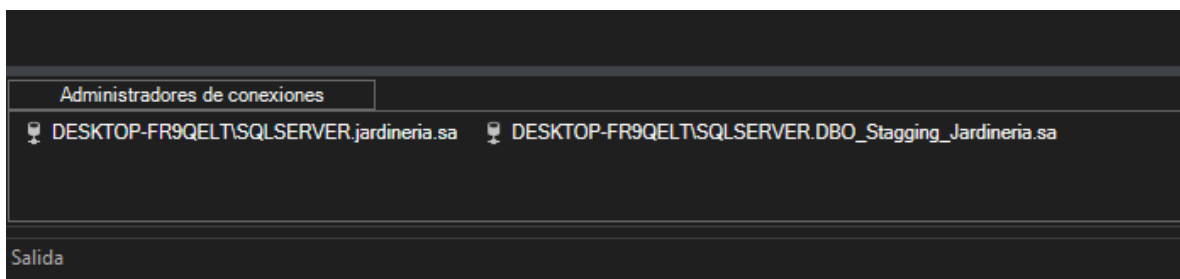


Ilustración 3 Conexión jardinería y staging

Se crean las dos conexiones necesarias para poder realizar el traslado de la información de las tablas.

Creación de base de datos *Stagging*

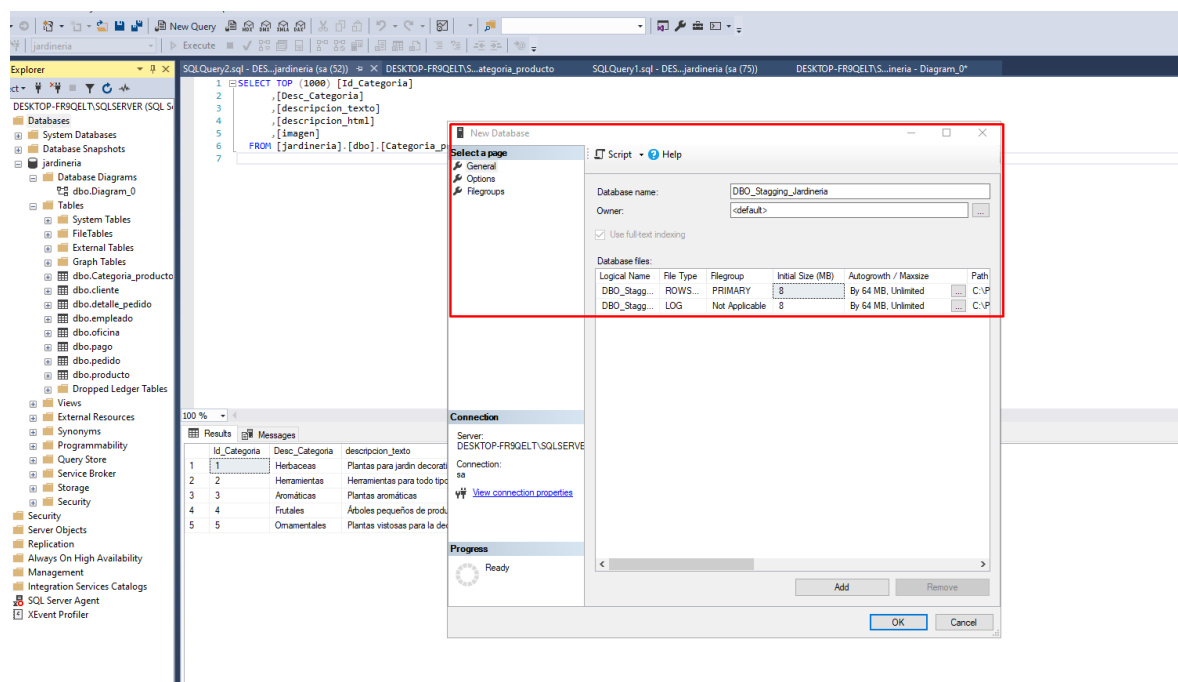


Ilustración 4 Creación de Base de datos Stagging

Se crea la base de datos **DBO_Stagging_Jardineria** para recepcionar los datos de la base de datos de jardinería, allí se creará cada consulta que permitirá traer los registros de cada tabla.

Comandos SQL

1. A partir de cada tabla entraremos al SMSS y procederemos a revisar los datos almacenados en el Jardinería para identificar cuáles son relevantes y cuáles se deben trasladar a la base de datos *Stagging*.

Tabla: Categoría_producto

En el caso de la tabla **categoría_producto** nos traemos todo menos **descripción_html** e **imagen** ya que cuentan con campos nulos.

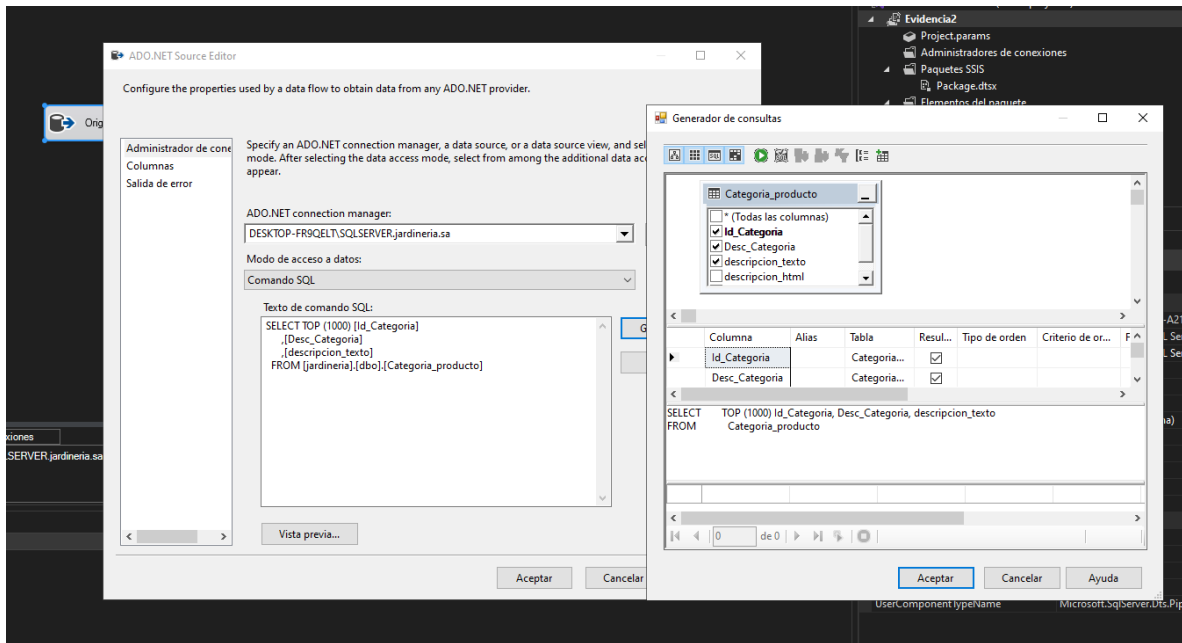
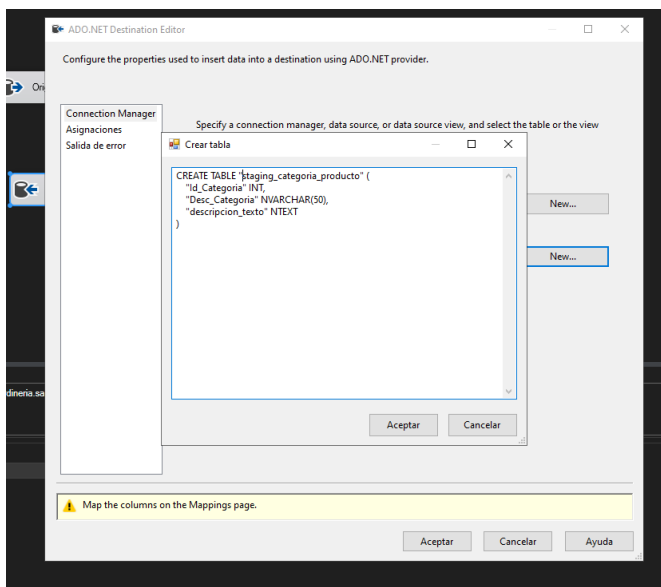


Ilustración 5 Categoría_Producto Comando SQL

Código SQL de tabla staging



```
CREATE TABLE
1 "staging_categoria_producto" (
2     "Id_Categoria" INT,
3     "Desc_Categoria" VARCHAR(50),
4     "descripcion_texto" TEXT
5 )
```

Ilustración 6 SQL creación tabla

Tabla: cliente

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

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
1	Goldfish Garden	Daniel G	Goldfish	5556501745	555901746	Falco Street 52 2 A	NULL	San Francisco	NULL	USA	24006	19	3000.00
2	Gardening Associates	Anne	Wright	5557410345	5557410346	Wall-e Avenue	NULL	Miami	USA	24010	19	6000.00	
3	Gerudo Valley	Link	Rhaze	555222123	555222123	Osaka Avenue #22	NULL	New York	NULL	USA	85495	22	12000.00
4	Tenda Garden	Alane	Tenda	5559123210	5559123211	Null Street #40	NULL	Miami	USA	696949	22	60000.00	
5	Lasas S.A.	Antonio	Lasas	3491540145	34914851312	C/Lagunas 15	NULL	Fuenteblanca	Madrid	Spain	28945	8	154310.00
6	Berapago	Jose	Berapago	615493721	916543672	C/centro segundo	Getafe	Madrid	Madrid	Spain	28942	11	20000.00
7	Club Golf Puerta del hemo	Paco	Lopez	62456810	919535678	C/sumero delgado	Madrid	Madrid	Madrid	Spain	28930	11	40000.00
8	Naturanga	Guillermo	Pungfu	69324790	916420955	C/napoli honda	Buñola	Madrid	Spain	28947	11	32000.00	
9	Dani Distribuciones	David	Serrano	679559001	916421756	C/razores	Fuenteblanca	Madrid	Madrid	Spain	28946	11	50000.00
10	Madriñeta de regios	Jose	Tacafio	655903045	916693215	C/Lagunas	Fuenteblanca	Madrid	Madrid	Spain	28943	11	20000.00
11	Lasas S.A.	Antonio	Lasas	3491540145	34914851312	C/Lagunas 15	NULL	Fuenteblanca	Madrid	Spain	28945	8	154310.00
12	Camunas jardines S.L.	Pablo	Camunas	34914877041	34914877041	C/Argemone 45	C/Procesos 2 118	San Lorenzo del Escorial	Madrid	Spain	28145	8	56481.00
13	Dardena S.A.	Juan	Rodriguez	34912453217	34912454764	C/Ribera York 74	NULL	Madrid	Madrid	Spain	28003	8	321000.00
14	Jardin de Flores	Javier	Villar	654865643	914538776	C/Olla 34	NULL	Madrid	Madrid	Spain	28950	30	40000.00
15	Roses Miami	Maria	Pedriguez	660595444	912486837	C/Lagunas24	NULL	Fuenteblanca	Madrid	Spain	28945	5	1500.00
16	Roses S.A.	Bautista	Fernandez	695794159	973453215	C/Las Salasulad	NULL	Norrome del valle	Barcelona	Spain	24586	5	3500.00
17	Naturangadi	Victoria	Cruz	612343529	916548735	Plaza Magallon 15	NULL	Madrid	Madrid	Spain	28011	30	5050.00
18	Golf S.A.	Luis	Martinez	915458762	912354475	C/Estancado	NULL	Santa cruz de Tenerife	Mas Canaries	Spain	38297	12	30000.00
19	Arench Golf Management SL	Ricco	Suarez	964493072	964493083	C/Letardo	NULL	Barcelona	Cataluña	Spain	12320	12	20000.00

Ilustración 7 Datos tabla cliente

Código SQL de tabla staggging

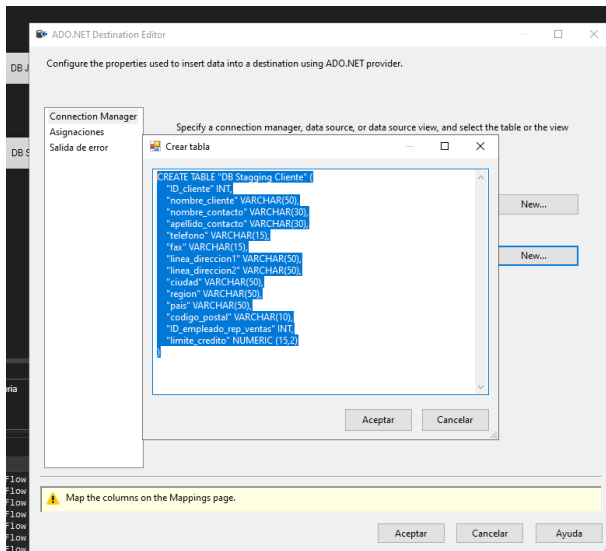


Ilustración 8 Codigo SQL Staggging

```
CREATE TABLE "DB Staggging
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)
)
```

Tabla: detalle_pedido

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

SQLQuery7.sql - DES...jardineria (sa (61))

```

1 SELECT TOP (1000) [ID_pedido]
2      ,[ID_producto]
3      ,[cantidad]
4      ,[precio_unidad]
5      ,[numero_linea]
6 FROM [jardineria].[dbo].[detalle_pedido]
7

```

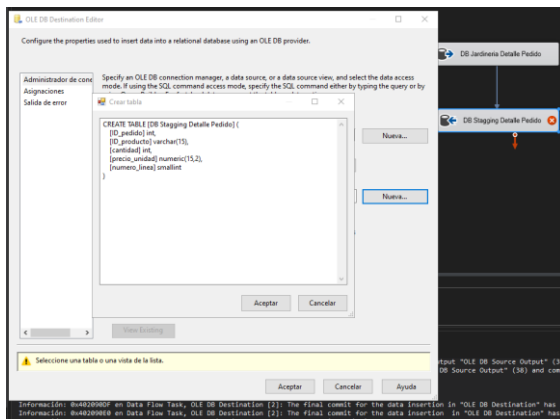
100 %

Results Messages

	ID_pedido	ID_producto	cantidad	precio_unidad	numero_linea
1	13	11679	5	14.00	1
2	17	11679	5	14.00	1
3	20	11679	14	14.00	1
4	32	11679	1	14.00	4
5	38	11679	5	14.00	1
6	54	11679	3	14.00	3
7	94	11679	12	14.00	1

Ilustración 9 Datos tabla detalle_pedido

Código SQL de tabla staging



```

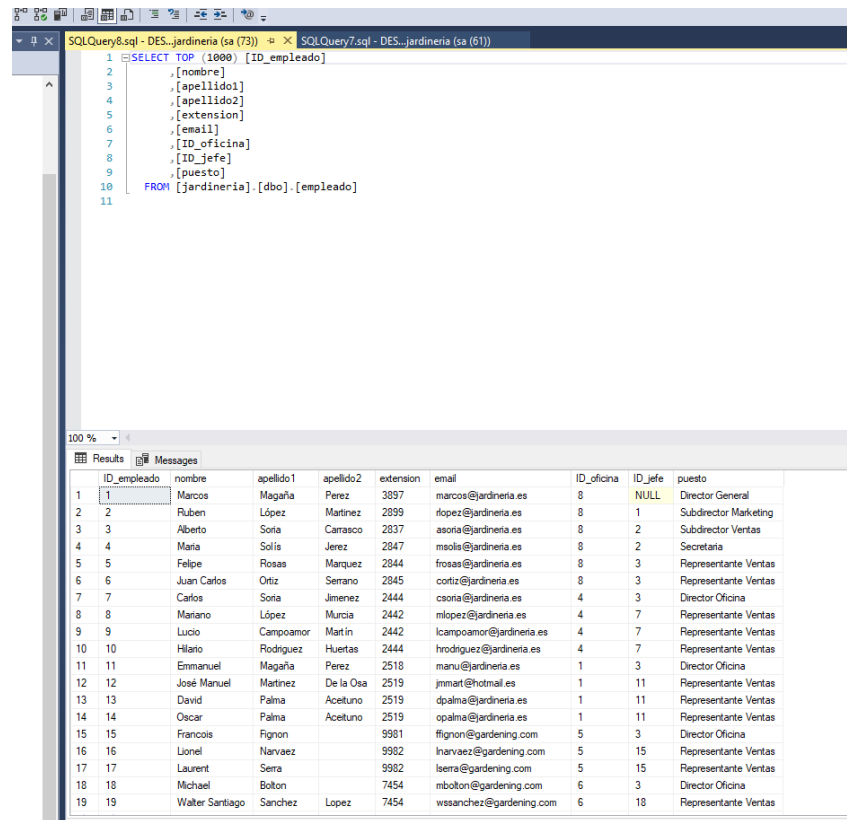
1 CREATE TABLE [DB Staging Detalle
2 Pedido] (
3     [ID_pedido] int,
4     [ID_producto] varchar(15),
5     [cantidad] int,
6     [precio_unidad] numeric(15,2),
7     [numero_linea] smallint
8 )

```

Ilustración 10 Código SQL Staging

Tabla: Empleado

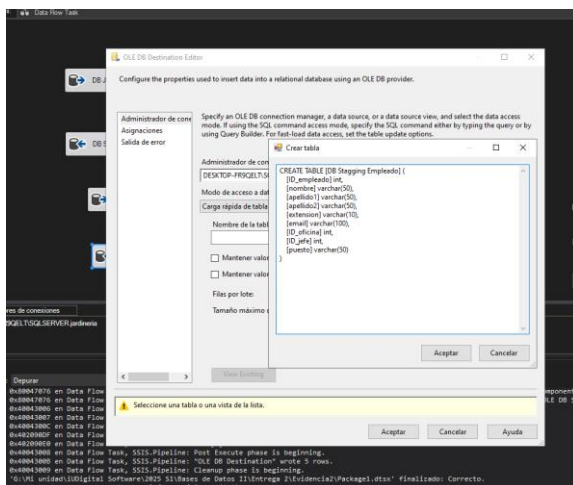
En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas



The screenshot shows a SQL query in the 'SQLQuery8.sql' window, which selects the top 1000 records from the 'Empleado' table. The results are displayed in a grid below the query editor.

ID_empleado	nombre	apellido1	apellido2	extension	email	ID_oficina	ID_jefe	puesto
1	Marcos	Magaña	Perez	3897	marcos@jardineria.es	8	NULL	Director General
2	Ruben	López	Martinez	2899	ropez@jardineria.es	8	1	Subdirector Marketing
3	Alberto	Soria	Carrasco	2837	asoria@jardineria.es	8	2	Subdirector Ventas
4	Maria	Solis	Jerez	2847	maolis@jardineria.es	8	2	Secretaria
5	Felipe	Rosas	Marquez	2844	frosas@jardineria.es	8	3	Representante Ventas
6	Juan Carlos	Ortiz	Serrano	2845	cortiz@jardineria.es	8	3	Representante Ventas
7	Carlos	Soria	Jimenez	2444	csoria@jardineria.es	4	3	Director Oficina
8	Mariano	López	Murcia	2442	mlopez@jardineria.es	4	7	Representante Ventas
9	Lucio	Campoamor	Martin	2442	lcampoamor@jardineria.es	4	7	Representante Ventas
10	Hilario	Rodriguez	Huetas	2444	hrodriguez@jardineria.es	4	7	Representante Ventas
11	Emmanuel	Magaña	Perez	2518	manu@jardineria.es	1	3	Director Oficina
12	José Manuel	Martinez	De la Osa	2519	jmmart@hotmail.es	1	11	Representante Ventas
13	David	Palma	Acetuno	2519	dpalma@jardineria.es	1	11	Representante Ventas
14	Oscar	Palma	Acetuno	2519	opalma@jardineria.es	1	11	Representante Ventas
15	Francois	Fignon		9981	ffignon@gardening.com	5	3	Director Oficina
16	Lionel	Narvaez		9982	lnarvaez@gardening.com	5	15	Representante Ventas
17	Laurent	Sera		9982	lserra@gardening.com	5	15	Representante Ventas
18	Michael	Bolton		7454	mbolton@gardening.com	6	3	Director Oficina
19	Walter Santiago	Sanchez	Lopez	7454	wssanchez@gardening.com	6	18	Representante Ventas

Ilustración 11 Tabla Empleado



```

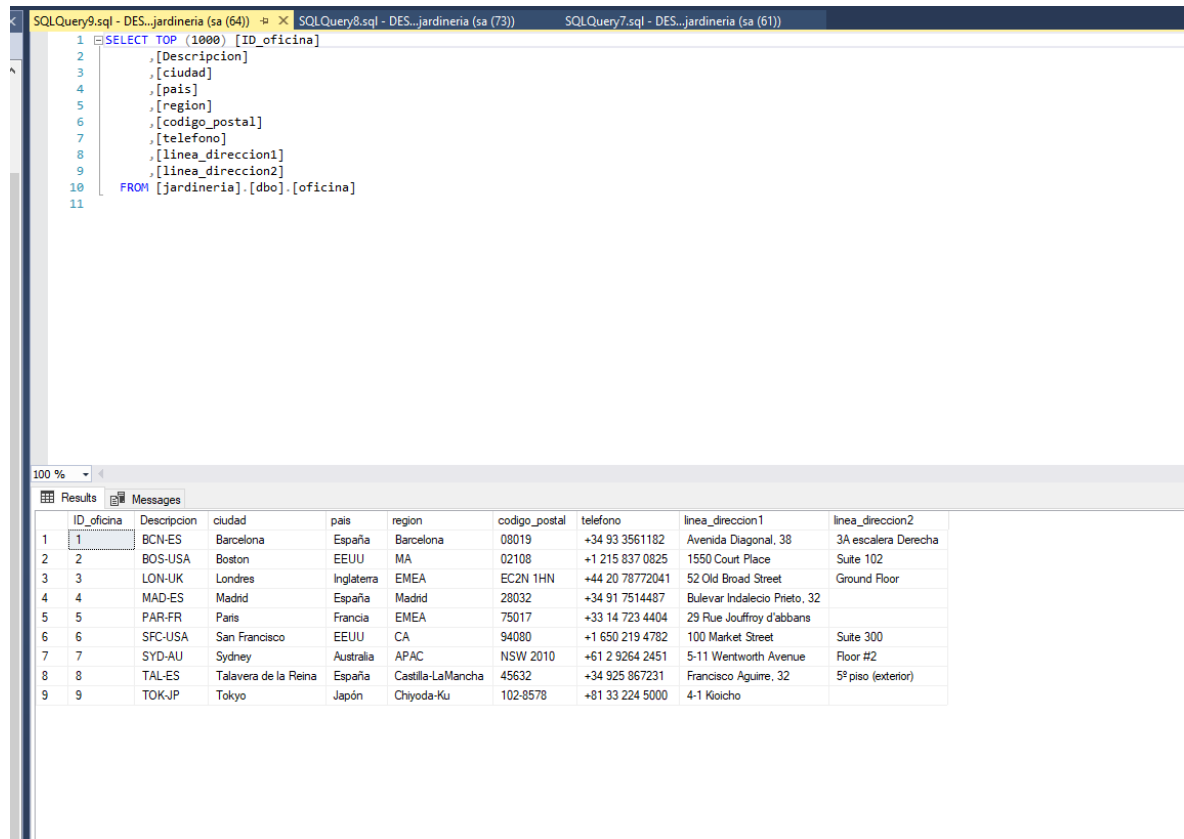
1 CREATE TABLE [DB Staging
2 Empleado] (
3     [ID_empleado] int,
4     [nombre] varchar(50),
5     [apellido1] varchar(50),
6     [apellido2] varchar(50),
7     [extension] varchar(10),
8     [email] varchar(100),
9     [ID_oficina] int,
10    [ID_jefe] int,
11    [puesto] varchar(50)
12 )

```

Ilustración 12 Codigo SQL Staging

Tabla: Oficina

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas



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

```
SELECT TOP (1000) [ID_oficina]
, [Descripcion]
, [ciudad]
, [pais]
, [region]
, [codigo_postal]
, [telefono]
, [linea_direccion1]
, [linea_direccion2]
FROM [jardineria].[dbo].[oficina]
```

The results window displays the following data:

ID_oficina	Descripcion	ciudad	pais	region	codigo_postal	telefono	linea_direccion1	linea_direccion2
1	BCN-ES	Barcelona	España	Barcelona	08019	+34 93 3561182	Avenida Diagonal, 38	3A escalera Derecha
2	BOS-USA	Boston	EEUU	MA	02108	+1 215 837 0825	1550 Court Place	Suite 102
3	LON-UK	Londres	Inglaterra	EMEA	EC2N 1HN	+44 20 78772041	52 Old Broad Street	Ground Floor
4	MAD-ES	Madrid	España	Madrid	28032	+34 91 7514487	Bulevar Indalecio Prieto, 32	
5	PAR-FR	Paris	Francia	EMEA	75017	+33 14 723 4404	29 Rue Joffroy d'abbans	
6	SFC-USA	San Francisco	EEUU	CA	94080	+1 650 219 4782	100 Market Street	Suite 300
7	SYD-AU	Sydney	Australia	APAC	NSW 2010	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2
8	TAL-ES	Talavera de la Reina	España	Castilla-LaMancha	45632	+34 925 867231	Francisco Aguirre, 32	5º piso (exterior)
9	TOK-JP	Tokyo	Japón	Chiyoda-Ku	102-8578	+81 33 224 5000	4-1 Kioicho	

Ilustración 13 Tabla Oficina

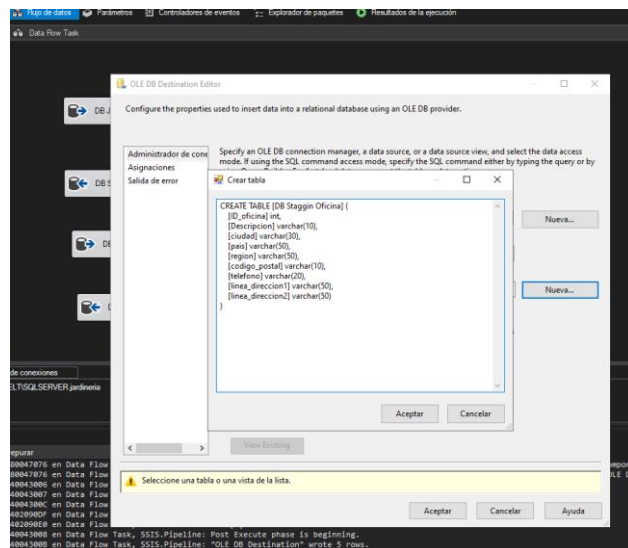


Ilustración 14 Codigo SQL Staggging

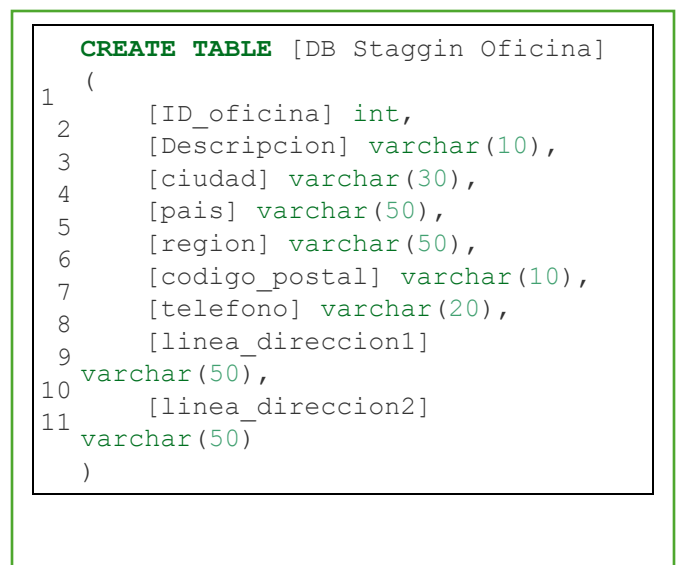
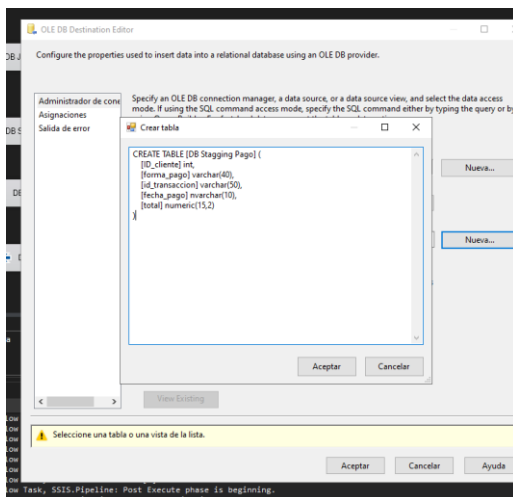


Tabla: Pago

En el caso de la tabla todas las columnas dado que no existen valores nulos en ellas

ID_cliente	forma_pago	id_transaccion	fecha_pago	total
1	PayPal	ak-std-000001	2008-11-10	2000.00
2	PayPal	ak-std-000002	2008-12-10	2000.00
3	PayPal	ak-std-000003	2009-01-16	5000.00
4	PayPal	ak-std-000004	2009-02-16	5000.00
5	PayPal	ak-std-000005	2009-02-19	526.00
6	PayPal	ak-std-000006	2007-01-08	20000.00
7	PayPal	ak-std-000007	2007-01-08	20000.00
8	PayPal	ak-std-000008	2007-01-08	20000.00
9	PayPal	ak-std-000009	2007-01-08	20000.00
10	PayPal	ak-std-000010	2007-01-08	1849.00
11	Transferencia	ak-std-000011	2006-01-18	23794.00
12	Cheque	ak-std-000012	2009-01-13	2390.00
13	PayPal	ak-std-000013	2009-01-06	529.00
14	PayPal	ak-std-000014	2008-08-04	2246.00
15	PayPal	ak-std-000015	2008-07-15	4160.00
16	PayPal	ak-std-000016	2009-01-15	2081.00
17	PayPal	ak-std-000035	2009-02-15	10000.00
18	PayPal	ak-std-000017	2009-02-16	4399.00
19	PayPal	ak-std-000018	2009-03-06	232.00

Ilustración 15Tabla Pago



```
1 CREATE TABLE [DB Staggin Pago] (
2     [ID_cliente] int,
3     [forma_pago] varchar(40),
4     [id_transaccion] varchar(50),
5     [fecha_pago] nvarchar(10),
6     [total] numeric(15,2)
7 )
```

Ilustración 16Codigo SQL Staggin

Tabla: Pedido

En el caso de la tabla pedido omitimos únicamente la columna comentario ya que no trae información importante para la tabla staging.

SQLQuery12.sql - DE_jardineria (sa (54)) SQLQuery10.sql - DE_jardineria (sa (80)) SQLQuery9.sql - DES_jardineria (sa (64))

```

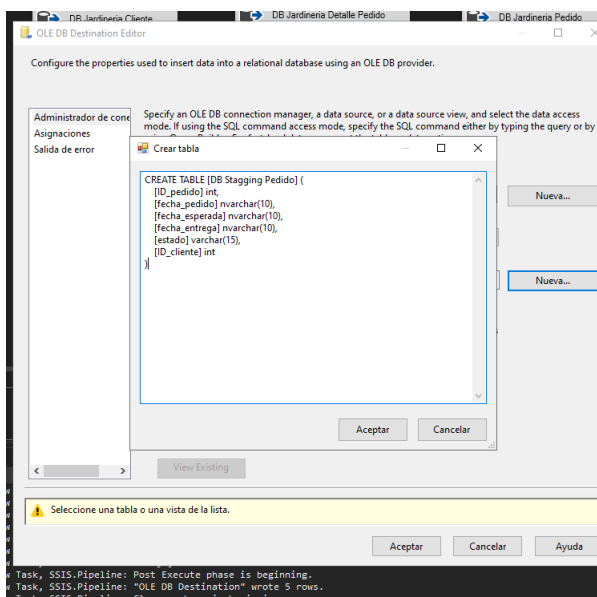
1 SELECT TOP (1000) [ID_pedido]
2     , [fecha_pedido]
3     , [fecha_esperada]
4     , [fecha_entrega]
5     , [estado]
6     , [comentarios]
7     , [ID_cliente]
8 FROM [jardineria].[dbo].[pedido]
9

```

ID_pedido	fecha_pedido	fecha_esperada	fecha_entrega	estado	comentarios	ID_cliente
1	2006-01-17	2006-01-19	2006-01-19	Entregado	Pagado a plazos	5
2	2007-10-23	2007-10-28	2007-10-26	Entregado	La entrega llegó antes de lo esperado	5
3	2008-06-20	2008-06-25	NULL	Rechazado	Límite de crédito superado	5
4	2009-01-20	2009-01-26	NULL	Pendiente	NULL	5
5	2008-11-09	2008-11-14	2008-11-14	Entregado	El cliente paga la mitad con tarjeta y la otra mita...	1
6	2008-12-22	2008-12-27	2008-12-28	Entregado	El cliente comprueba la integridad del paquete, ...	1
7	2009-01-15	2009-01-20	NULL	Pendiente	El cliente llama para confirmar la fecha - Espere...	3
8	2009-01-20	2009-01-27	NULL	Pendiente	El cliente requiere que el pedido se le entregue ...	1
9	2009-01-22	2009-01-27	NULL	Pendiente	El cliente requiere que el pedido se le entregue ...	1
10	2009-01-12	2009-01-14	2009-01-15	Entregado	NULL	7
11	2009-01-02	2009-01-02	NULL	Rechazado	mal pago	7
12	2009-01-09	2009-01-12	2009-01-11	Entregado	NULL	7
13	2009-01-06	2009-01-07	2009-01-15	Entregado	NULL	7
14	2009-01-08	2009-01-09	2009-01-11	Entregado	mal estado	7
15	2009-01-05	2009-01-06	2009-01-07	Entregado	NULL	9
16	2009-01-18	2009-02-12	NULL	Pendiente	entregar en murcia	9
17	2009-01-20	2009-02-15	NULL	Pendiente	NULL	9
18	2009-01-09	2009-01-09	2009-01-09	Rechazado	mal pago	9
19	2009-01-11	2009-01-11	2009-01-13	Entregado	NULL	9

Query executed successfully.

Ilustración 17 Tabla Pedido



```

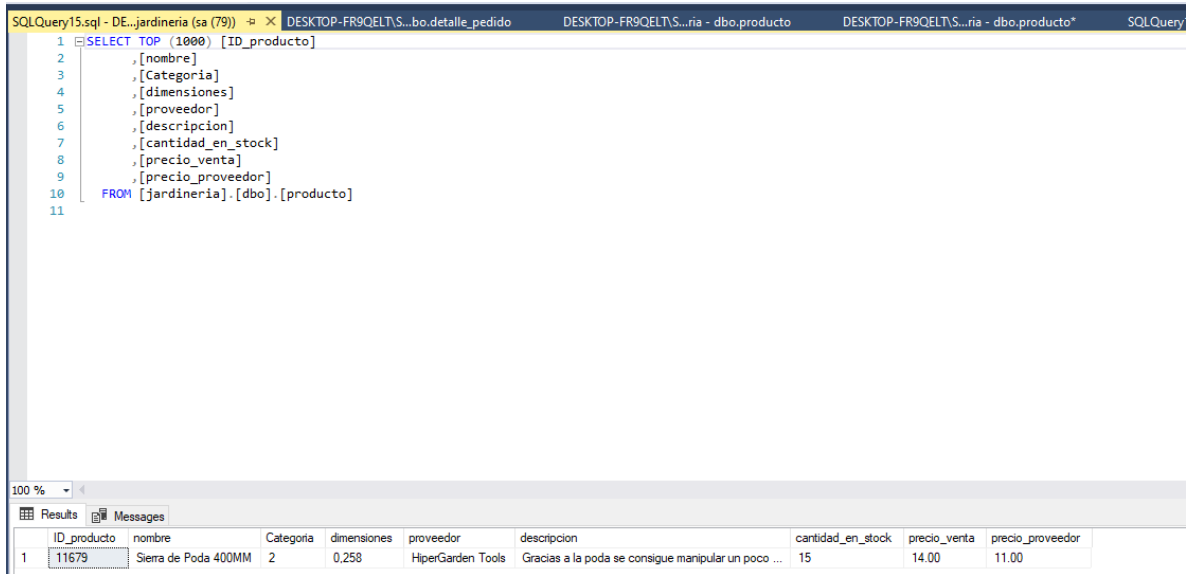
1 CREATE TABLE [DB Staging Pedido]
2 (
3     [ID_pedido] int,
4     [fecha_pedido] nvarchar(10),
5     [fecha_esperada] nvarchar(10),
6     [fecha_entrega] nvarchar(10),
7     [estado] varchar(15),
8     [ID_cliente] int
9 )

```

Ilustración 18 Código SQL Staging

Table: Producto

Se omite la columna descripción y se importan las demás columnas



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

```

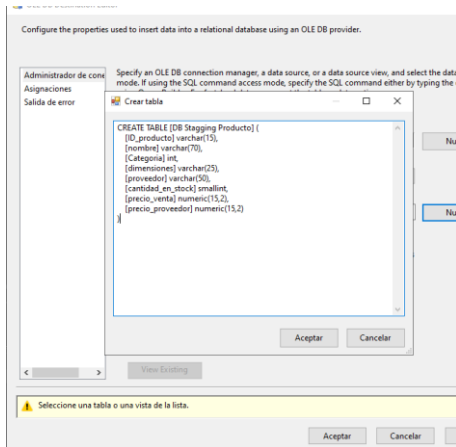
1 SELECT TOP (1000) [ID_producto]
2     , [nombre]
3     , [Categoria]
4     , [dimensiones]
5     , [proveedor]
6     , [descripcion]
7     , [cantidad_en_stock]
8     , [precio_venta]
9     , [precio_proveedor]
10 FROM [jardineria].[dbo].[producto]
11

```

The Results pane shows the following data:

ID_producto	nombre	Categoria	dimensiones	proveedor	descripcion	cantidad_en_stock	precio_venta	precio_proveedor
11679	Sierra de Poda 400MM	2	0.258	HiperGarden Tools	Gracias a la poda se consigue manipular un poco ...	15	14.00	11.00

Ilustración 19Tabla SQL Pedido



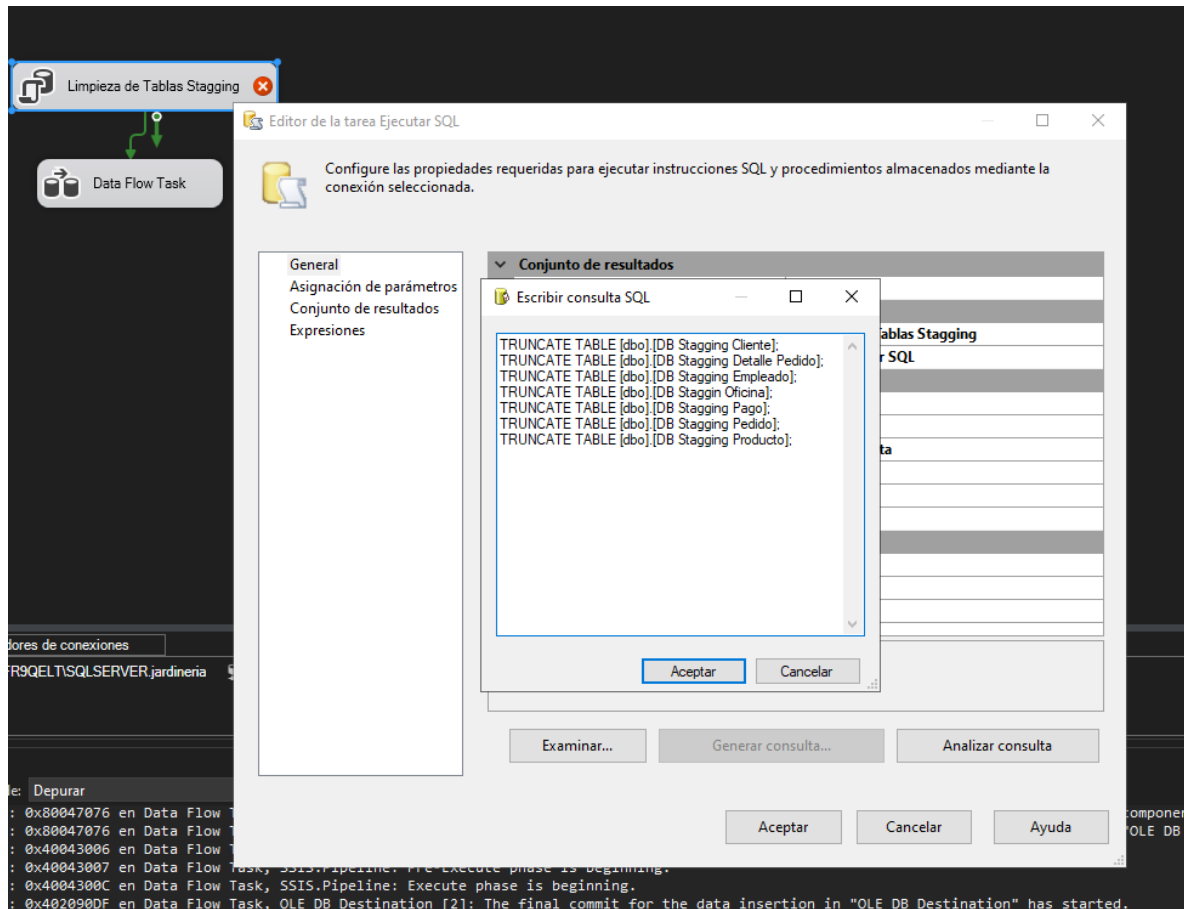
```

1 CREATE TABLE [DB Staging Producto] (
2     [ID_producto] varchar(15),
3     [nombre] varchar(70),
4     [Categoria] int,
5     [dimensiones] varchar(25),
6     [proveedor] varchar(50),
7     [cantidad_en_stock] smallint,
8     [precio_venta] numeric(15,2),
9     [precio_proveedor] numeric(15,2)
10 )

```

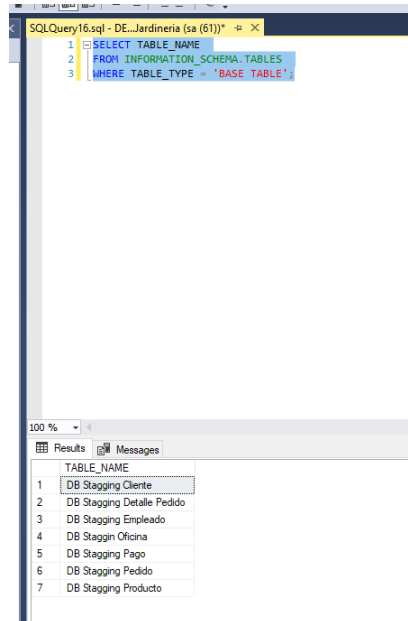
Ilustración 20Codigo SQL Staging

Código SQL para limpieza de Tablas *Stagging*



```
1 TRUNCATE TABLE [dbo].[DB Stagging Cliente];
2 TRUNCATE TABLE [dbo].[DB Stagging Detalle Pedido];
3 TRUNCATE TABLE [dbo].[DB Stagging Empleado];
4 TRUNCATE TABLE [dbo].[DB Staggin Oficina];
5 TRUNCATE TABLE [dbo].[DB Stagging Pago];
6 TRUNCATE TABLE [dbo].[DB Stagging Pedido];
7 TRUNCATE TABLE [dbo].[DB Stagging Producto];
```


Optimización de tablas



Obtenemos el nombre de todas las tablas para mejorarlas como un proceso ETL usando SISS INCLUIAMOS campos como **"Fecha de carga"** y **"Estado de carga"** para rastrear cuándo y cómo se cargaron los datos en la base de datos a cada tabla.

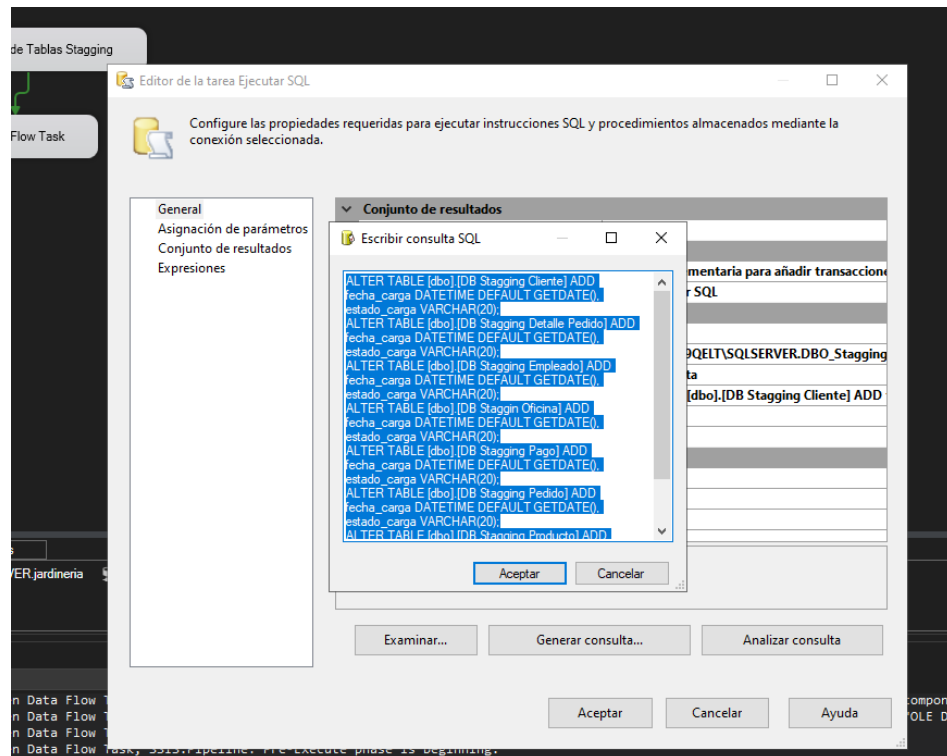


Ilustración 21 Tarea complementaria manual

Código SQL para complementar las tablas

```

ALTER TABLE [dbo].[DB Staging Cliente] ADD fecha_carga DATETIME
DEFAULT GETDATE(), estado_carga VARCHAR(20);
ALTER TABLE [dbo].[DB Staging Detalle Pedido] ADD fecha_carga
DATETIME DEFAULT GETDATE(), estado_carga VARCHAR(20);
1 ALTER TABLE [dbo].[DB Staging Empleado] ADD fecha_carga DATETIME
2 DEFAULT GETDATE(), estado_carga VARCHAR(20);
3 ALTER TABLE [dbo].[DB Staging Oficina] ADD fecha_carga DATETIME
4 DEFAULT GETDATE(), estado_carga VARCHAR(20);
5 ALTER TABLE [dbo].[DB Staging Pago] ADD fecha_carga DATETIME DEFAULT
6 GETDATE(), estado_carga VARCHAR(20);
7 ALTER TABLE [dbo].[DB Staging Pedido] ADD fecha_carga DATETIME
DEFAULT GETDATE(), estado_carga VARCHAR(20);
ALTER TABLE [dbo].[DB Staging Producto] ADD fecha_carga DATETIME
DEFAULT GETDATE(), estado_carga VARCHAR(20);

```

Resultados

Estado del flujo de tareas

The screenshot displays the SSDT interface with a Data Flow Task (DFT) diagram. The diagram shows a sequence of data flows from source tables to staging tables and then to target tables. The source tables are: DB Jardinera Categoria_Producto, DB Jardinera Cliente, DB Jardinera Detalle Pedido, DB Jardinera Pedido, DB Jardinera Empleado, DB Jardinera Oficina, DB Jardinera Pago, and DB Jardinera Producto. These are mapped to staging tables: DB Staging Categoria_Producto, DB Staging Cliente, DB Staging Detalle Pedido, DB Staging Pedido, DB Staging Empleado, DB Staging Oficina, DB Staging Pago, and DB Staging Producto. The target tables are: DB Staging Categoria_Producto, DB Staging Cliente, DB Staging Detalle Pedido, DB Staging Pedido, DB Staging Empleado, DB Staging Oficina, DB Staging Pago, and DB Staging Producto. The execution results pane at the bottom shows the following output:

```

Salida
Mostrar salida de: Depurar
Advertencia: 0x80047076 en Data Flow Task, SSIS.Pipeline: The output column "descripcion_html" (42) on output "OLE DB Source Output" (38) and component "OLE DB Source" (38) is not mapped to any column in the destination.
Advertencia: 0x80047076 en Data Flow Task, SSIS.Pipeline: The output column "imagen" (43) on output "OLE DB Source Output" (38) and component "OLE DB Source" (38) is not mapped to any column in the destination.
Información: 0x40043006 en Data Flow Task, SSIS.Pipeline: Prepare for Execute phase is beginning.
Información: 0x40043007 en Data Flow Task, SSIS.Pipeline: Pre-Execute phase is beginning.
Información: 0x4004300C en Data Flow Task, SSIS.Pipeline: Execute phase is beginning.
Información: 0x4020900F en Data Flow Task, OLE DB Destination [2]: The final commit for the data insertion in "OLE DB Destination" has started.
Información: 0x4020900F en Data Flow Task, OLE DB Destination [2]: The final commit for the data insertion in "OLE DB Destination" has ended.
Información: 0x40043008 en Data Flow Task, SSIS.Pipeline: Post-Execute phase is beginning.
Información: 0x40043008 en Data Flow Task, SSIS.Pipeline: "OLE DB Destination" wrote 5 rows.
Información: 0x40043009 en Data Flow Task, SSIS.Pipeline: Cleanup phase is beginning.
Paquete SSIS "G:\VH\unidadIU\Digital Software\0025 SQL Bases de Datos I1\Intrega 2\Evidencia2\Package1.dtsx" #finalizado: Correcto.
El programa "1620 DtsDebugHost.exe: DTS" terminó con código 0 (0x0).

```



Finalmente validamos los datos

SQLQuery17.sql - DE...Jardineria (sa (73))

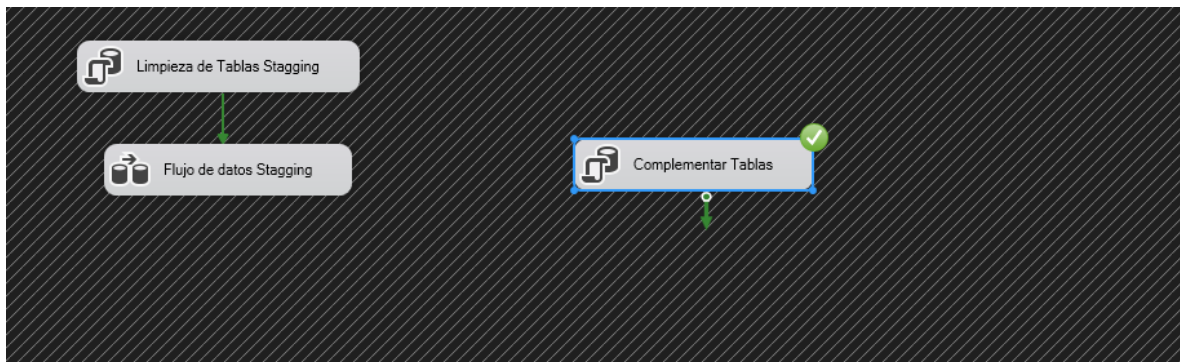
```

1 SELECT TOP (1000) [Id_Categoria]
2 , [Desc_Categoria]
3 , [descripcion_texto]
4 FROM [DBO_Staging_Jardineria].[dbo].[DB_Staging_Categoria_Producto]
5
  
```

Results

	Id_Categoria	Desc_Categoria	descripcion_texto
1	1	Herbaceas	Plantas para jardín decorativas
2	2	Herramientas	Herramientas para todo tipo de acción
3	3	Aromáticas	Plantas aromáticas
4	4	Frutales	Árboles pequeños de producción frutal
5	5	Ornamentales	Plantas vistosas para la decoración del jardín

Complementamos los datos



De esta manera la próxima vez que se haga un cargue se ejecutará los triggers adecaudos para registrar los datos en el momento indicado

SQLQuery18.sql - DE...Jardineria (sa (66)) SQLQuery17.sql - DE...Jardineria (sa (73)) SQLQuery16.sql - DE...Jardineria (sa (61))*

```

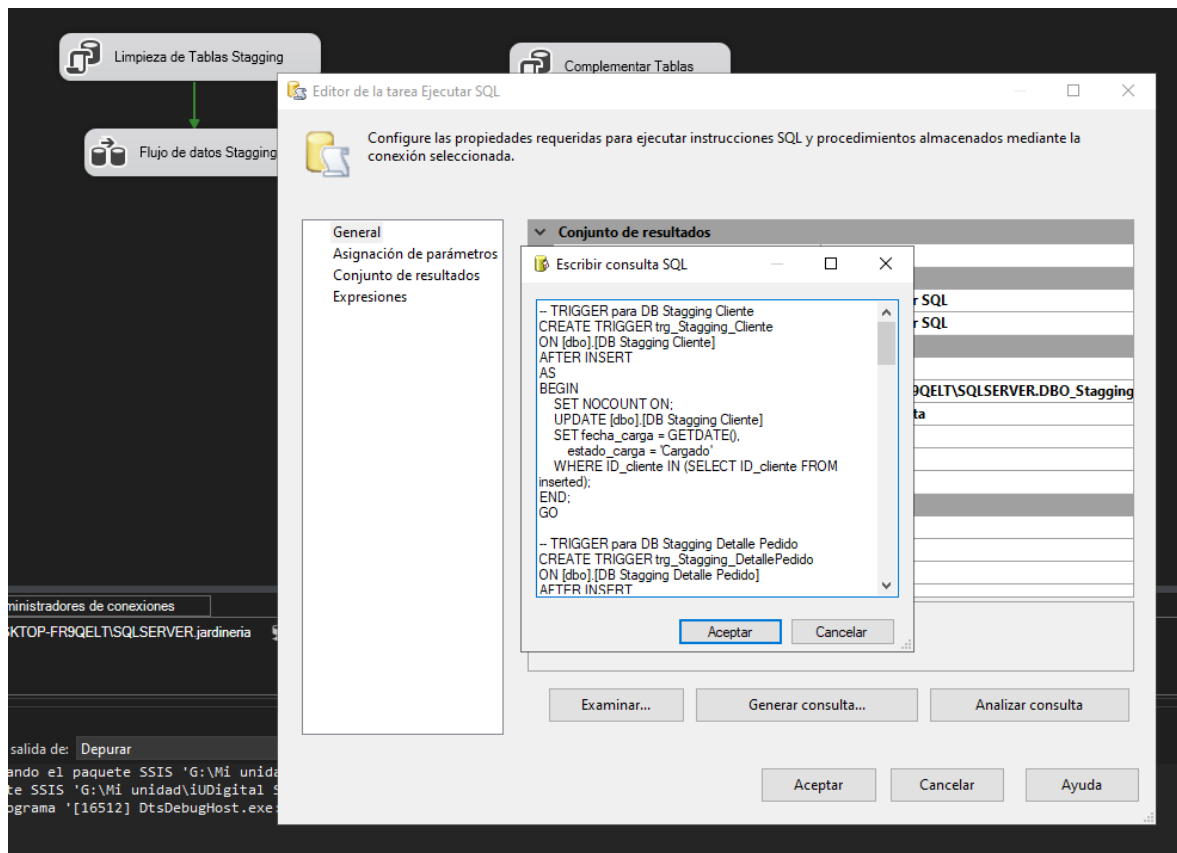
1 SELECT TOP (1000) [ID_oficina]
2     ,[Descripcion]
3     ,[ciudad]
4     ,[pais]
5     ,[region]
6     ,[codigo_postal]
7     ,[telefono]
8     ,[linea_direccion1]
9     ,[linea_direccion2]
10    ,[fecha_carga]
11    ,[estado_carga]
12 FROM [DBO_Staging_Jardineria].[dbo].[DB_Staggin_Oficina]
13

```

100 %

ID_oficina	Descripcion	ciudad	pais	region	codigo_postal	telefono	linea_direccion1	linea_direccion2	fecha_carga	estado_carga
1	BCN-ES	Barcelona	España	Barcelona	08019	+34 93 3561182	Avenida Diagonal, 38	3A escalera Derecha	NULL	NULL
2	BOS-USA	Boston	EEUU	MA	02108	+1 215 837 0825	1550 Court Place	Suite 102	NULL	NULL
3	LON-UK	Londres	Inglaterra	EMEA	EC2N 1HN	+44 20 78772041	52 Old Broad Street	Ground Floor	NULL	NULL
4	MAD-ES	Madrid	España	Madrid	28032	+34 91 7514487	Bulevar Indalecio Prieto, 32		NULL	NULL
5	PAR-FR	Paris	Francia	EMEA	75017	+33 14 723 4404	29 Rue Jouffroy d'abbans		NULL	NULL
6	SFC-USA	San Francisco	EEUU	CA	94080	+1 650 219 4782	100 Market Street	Suite 300	NULL	NULL
7	SYD-AU	Sydney	Australia	APAC	NSW 2010	+61 2 9264 2451	5-11 Wentworth Avenue	Floor #2	NULL	NULL
8	TAL-ES	Talavera de la Reina	España	Castilla-LaMancha	45632	+34 925 867231	Francisco Aguirre, 32	5º piso (exterior)	NULL	NULL
9	TOK-JP	Tokyo	Japón	Chiyoda-Ku	102-8578	+81 33 224 5000	4-1 Koicho		NULL	NULL

Lo único que queda pendiente es añadir un trigger para ejecutar los estados de cargue y registrar el timestamp



Codigo SQL Triggers

```

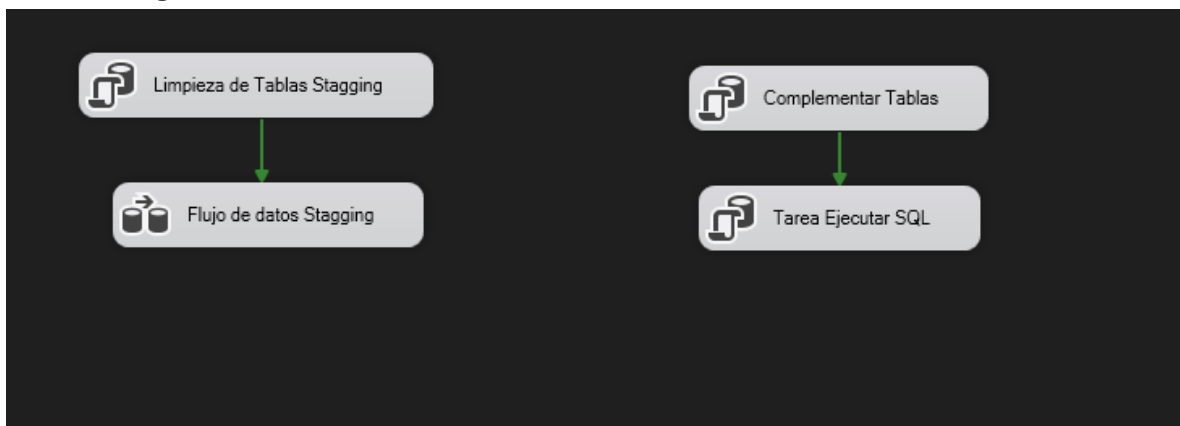
1  -- TRIGGER para DB Stagging Cliente
2  CREATE TRIGGER trg_Stagging_Cliente
3  ON [dbo].[DB Stagging Cliente]
4  AFTER INSERT
5  AS
6  BEGIN
7      SET NOCOUNT ON;
8      UPDATE [dbo].[DB Stagging Cliente]
9      SET fecha_carga = GETDATE(),
10         estado_carga = 'Cargado'
11      WHERE ID_cliente IN (SELECT ID_cliente FROM inserted);
12 END;
13 GO
14
15 -- TRIGGER para DB Stagging Detalle Pedido
16 CREATE TRIGGER trg_Stagging_DetallePedido
17 ON [dbo].[DB Stagging Detalle Pedido]
18 AFTER INSERT
19 AS

```

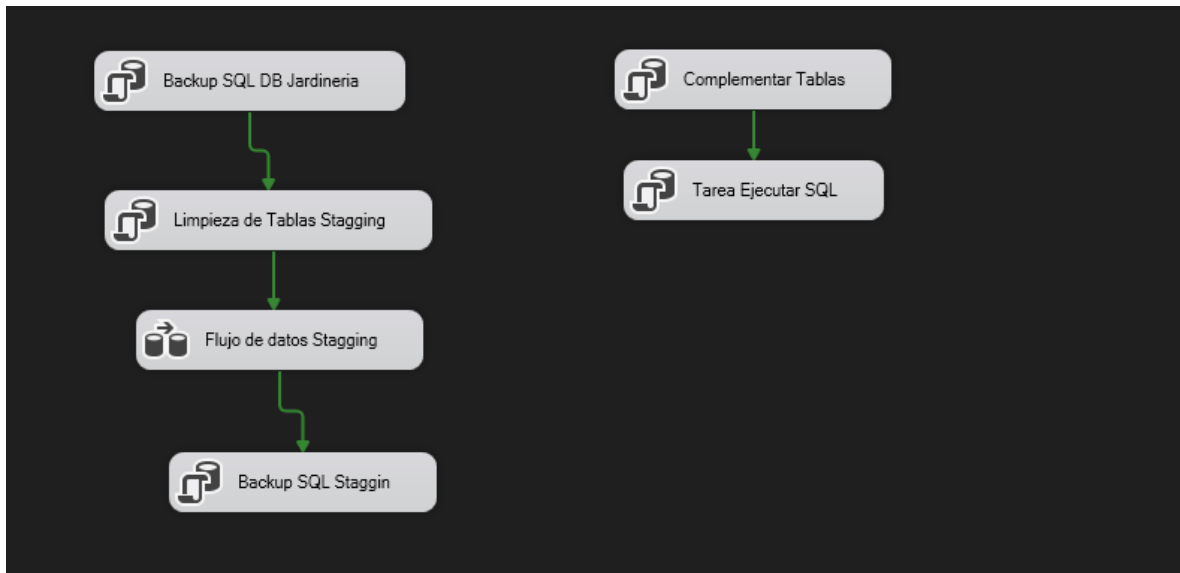
```
20 BEGIN
21     SET NOCOUNT ON;
22     UPDATE [dbo].[DB Stagging Detalle Pedido]
23     SET fecha_carga = GETDATE(),
24         estado_carga = 'Cargado'
25     WHERE ID_pedido IN (SELECT ID_pedido FROM inserted);
26 END;
27 GO
28
29 -- TRIGGER para DB Stagging Empleado
30 CREATE TRIGGER trg_Stagging_Empleado
31 ON [dbo].[DB Stagging Empleado]
32 AFTER INSERT
33 AS
34 BEGIN
35     SET NOCOUNT ON;
36     UPDATE [dbo].[DB Stagging Empleado]
37     SET fecha_carga = GETDATE(),
38         estado_carga = 'Cargado'
39     WHERE ID_empleado IN (SELECT ID_empleado FROM inserted);
40 END;
41 GO
42
43 -- TRIGGER para DB Stagging Oficina
44 CREATE TRIGGER trg_Stagging_Oficina
45 ON [dbo].[DB Staggin Oficina]
46 AFTER INSERT
47 AS
48 BEGIN
49     SET NOCOUNT ON;
50     UPDATE [dbo].[DB Staggin Oficina]
51     SET fecha_carga = GETDATE(),
52         estado_carga = 'Cargado'
53     WHERE ID_oficina IN (SELECT ID_oficina FROM inserted);
54 END;
55 GO
56
57 -- TRIGGER para DB Stagging Pago
58 CREATE TRIGGER trg_Stagging_Pago
59 ON [dbo].[DB Stagging Pago]
60 AFTER INSERT
61 AS
62 BEGIN
63     SET NOCOUNT ON;
64     UPDATE [dbo].[DB Stagging Pago]
65     SET fecha_carga = GETDATE(),
66         estado_carga = 'Cargado'
67     WHERE ID_cliente IN (SELECT ID_cliente FROM inserted);
68 END;
69 GO
70
```

```
71 -- TRIGGER para DB Stagging Pedido
72 CREATE TRIGGER trg_Stagging_Pedido
73 ON [dbo].[DB Stagging Pedido]
74 AFTER INSERT
75 AS
76 BEGIN
77     SET NOCOUNT ON;
78     UPDATE [dbo].[DB Stagging Pedido]
79     SET fecha_carga = GETDATE(),
80         estado_carga = 'Cargado'
81     WHERE ID_pedido IN (SELECT ID_pedido FROM inserted);
82 END;
83 GO
84
85 -- TRIGGER para DB Stagging Producto
86 CREATE TRIGGER trg_Stagging_Producto
87 ON [dbo].[DB Stagging Producto]
88 AFTER INSERT
89 AS
90 BEGIN
91     SET NOCOUNT ON;
92     UPDATE [dbo].[DB Stagging Producto]
93     SET fecha_carga = GETDATE(),
94         estado_carga = 'Cargado'
95     WHERE ID_producto IN (SELECT ID_producto FROM inserted);
96 END;
97 GO
```

Resultado general



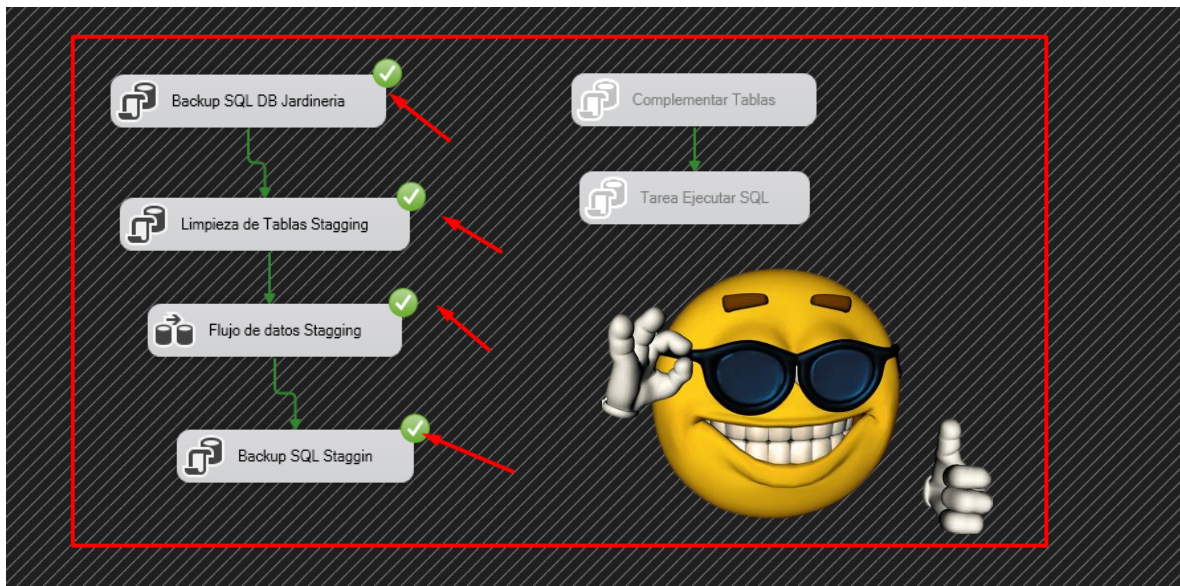
Tarea de Generar Backups



Se añade dos tareas correspondientes para generar los backups de las dos bases de datos, como medida preventiva

Codigo SQL

```
1 -- Backup de la base de datos principal
2 BACKUP DATABASE jardineria
3 TO DISK = 'C:\Backups\jardineria_backup.bak'
4 WITH FORMAT, INIT,
5 NAME = 'Backup Completo de jardineria',
6 SKIP, NOREWIND, NOUNLOAD, STATS = 10;
7 GO
8
9 -- Backup de la base de datos de staging
10 BACKUP DATABASE DBO_Stagging_Jardineria
11 TO DISK = 'C:\Backups\jardineria_staging_backup.bak'
12 WITH FORMAT, INIT,
13 NAME = 'Backup Completo de jardineria_staging',
14 SKIP, NOREWIND, NOUNLOAD, STATS = 10;
15 GO
```


Resultados satisfactorios

Conclusiones

- La implementación de una base de datos Staging permite manejar la información de manera más organizada y controlada antes de ser procesada.
- La migración de datos debe realizarse mediante consultas optimizadas para evitar sobrecarga en los sistemas y asegurar la correcta transferencia.
- Es fundamental validar la cantidad de registros en ambas bases de datos para garantizar que no se pierda información durante el proceso de carga.
- La inclusión de campos como **fecha de carga** y **estado de carga** permite un mejor monitoreo y control del proceso ETL.
- La realización de respaldos antes y después del proceso de migración es una práctica clave para mitigar riesgos de pérdida de información.

Bibliografía

Microsoft. (s.f.). *CREATE TRIGGER (Transact-SQL)*. Microsoft Learn. Recuperado el 16 de marzo de 2025, de <https://learn.microsoft.com/es-es/sql/t-sql/statements/create-trigger-transact-sql>

Aprendiendo SQL Server. (2013). *Programar triggers correctamente: ¿Qué hacer y qué no hacer?*. Blog Aprendiendo SQL Server. Recuperado el 16 de marzo de

2025, de <https://aprendiendosqlserver.blogspot.com/2013/09/programar-triggers-correctamente-que.html>

Blog SSIS Castellano. (2015). *Implementación y carga de dimensiones y tablas de hechos en SSIS*. Recuperado el 16 de marzo de 2025, de https://blogssiscastellano.blogspot.com/2015/07/implementacion-y-carga-de-dimensiones-y_65.html

Certia. (s.f.). *Seis razones por las que usar staging en proyectos ETL*. Recuperado el 16 de marzo de 2025, de <https://www.certia.net/seis-razones-por-las-que-usar-staging>