



Estudiante:

Anderson Gaviria Bedoya

Robert Andrés Castillo Gaviria

Oscar Javier García García

María Fernanda Vásquez Montiel

Matrícula:

PREICA2502B010064

Unidad 2:

Base de datos staging

Nombre Tutor:

Antonio Jesús Valderrama

Fecha de elaboración:

14 de septiembre de 2025

Introducción

En los sistemas informáticos modernos, la gestión y análisis de datos son muy importantes para la toma de decisiones estratégicas. La base de datos *Jardinería* contiene información de carácter transaccional relacionada con clientes, productos, oficinas, empleados y pedidos. Sin embargo, para fines analíticos, resulta complejo trabajar directamente con estos datos, debido a su dispersión y a la falta de integración en un entorno diseñado para explotación analítica. Por ello, se requiere la construcción de una base de datos *Staging*, que sirva como área intermedia de almacenamiento y preparación de datos en un proceso ETL (Extracción, Transformación y Carga).

Objetivo general

Diseñar e implementar la base de datos *Staging* a partir de los datos de *Jardinería*, garantizando la correcta integración y preparación para futuros procesos de análisis.

Objetivos específicos

- Analizar las tablas y atributos relevantes de la base de datos *Jardinería*.
- Diseñar la estructura de la base de datos *Staging*.
- Construir y ejecutar las consultas que permitan trasladar los datos desde *Jardinería*.
- Validar que los registros se almacenen de forma correcta en *Staging*.
- Generar respaldos (BK) de ambas bases de datos

Planteamiento del problema

La base de datos *Jardinería* fue diseñada principalmente con un enfoque transaccional, lo que significa que está optimizada para registrar operaciones del día a día como ventas, pedidos, pagos y gestión de clientes. Sin embargo, cuando intentamos usar esa información para análisis más avanzados, nos encontramos con varias dificultades.

En primer lugar, los datos están distribuidos en múltiples tablas, lo que hace que obtener reportes completos sea un proceso lento y complejo. Además, la base original no siempre está estructurada de la mejor manera para integrarse en un sistema de análisis, ya que contiene redundancias y relaciones que complican la extracción de información.

Otro problema es que la base de datos no fue pensada para la preparación previa de datos (limpieza, estandarización o transformación), lo cual se vuelve un obstáculo al momento de trasladar la información hacia un sistema de Business Intelligence o un Data Warehouse.

Por estas razones surge la necesidad de crear un staging, que nos sirva como un entorno intermedio para organizar y depurar los datos antes de llevarlos a un modelo analítico. Con este staging podemos centralizar la información, simplificar las estructuras y garantizar que los datos estén listos para futuros procesos de análisis.

Análisis del problema

Al revisar la base de datos Jardinería y los procesos que se quieren cubrir, identificamos varias situaciones que dificultan trabajar directamente con ella:

1. **Dispersión de datos:** la información de clientes, pedidos, pagos y productos está repartida en muchas tablas relacionadas. Esto genera consultas complejas y lentas, especialmente cuando se quieren hacer reportes integrales.
2. **Estructura orientada a transacciones:** el diseño de Jardinería se centra en registrar operaciones, pero no en preparar los datos para análisis. Por ejemplo, atributos como comentarios en pedidos o direcciones con varias líneas no siempre son útiles en un entorno analítico.
3. **Dependencias y restricciones:** la gran cantidad de claves foráneas puede causar problemas al cargar o transformar datos, lo que complica los procesos de integración hacia un sistema de análisis.

4. **Necesidad de limpieza y estandarización:** algunos campos (como teléfonos, direcciones o correos) pueden tener inconsistencias en su formato, lo cual representa un reto si se quiere hacer minería de datos o análisis comparativo.
5. **Falta de un área intermedia:** al no existir un staging, los datos se moverían directamente de la base transaccional al análisis, lo que incrementa el riesgo de errores, pérdida de calidad y menor rendimiento.

En conclusión, el problema central radica en que la base de datos Jardinería, aunque es eficiente para operaciones diarias, no resulta adecuada para análisis ni para integrarse fácilmente a un *data warehouse*. Por eso, el staging aparece como la mejor solución para organizar, simplificar y preparar la información antes de llevarla a un entorno analítico.

Propuesta de solución

Para dar respuesta a los problemas identificados, realizamos un análisis detallado de las tablas y atributos de la base de datos Jardinería, con el fin de seleccionar la información más relevante y simplificar su estructura en la base de datos Staging.

En primer lugar, identificamos que la base Jardinería tenía un nivel de detalle muy alto, incluyendo campos que no eran necesarios para el proceso de análisis, como direcciones adicionales, comentarios de pedidos o descripciones largas de productos. Estos datos, aunque útiles en un sistema transaccional, no aportaban valor dentro de un entorno de preparación para análisis.

Después, se diseñó la base de datos Staging tomando únicamente los atributos que consideramos más importantes para mantener la trazabilidad de los clientes, empleados, productos, oficinas, pedidos y pagos. Con este enfoque se redujo la complejidad del modelo y se evitó la sobrecarga de información.

El traslado de los datos se pensó de manera que cada tabla del staging correspondiera a una tabla de Jardinería, pero con menos columnas y sin dependencias innecesarias. Por ejemplo:

- **Oficina:** se eliminaron los campos de dirección secundaria, quedando solo ciudad, país, código postal y teléfono.
- **Empleado:** se redujeron los atributos a los esenciales (nombre, apellidos, oficina, jefe y puesto), dejando de lado extensiones o correos.
- **Cliente:** se conservaron los datos de identificación, contacto, ubicación y límite de crédito, omitiendo campos secundarios como fax o direcciones adicionales.
- **Producto:** se mantuvieron nombre, gama, dimensiones, proveedor y precio de venta, dejando de lado las descripciones largas y el precio del proveedor.
- **Pedido y detalle pedido:** se simplificaron eliminando comentarios y fechas intermedias, conservando únicamente lo necesario para identificar al cliente, las fechas principales y el estado.
- **Pago:** se dejó con la información esencial: cliente, forma de pago, fecha y total.

Este análisis permitió depurar la información y trasladarla al staging de forma más ordenada y lista para futuras transformaciones. Con la estructura diseñada, ahora es más sencillo realizar consultas, integrar los datos y preparar la carga hacia un *data* warehouse o cualquier otro sistema de análisis.

esquema tablas db staging datos más relevantes

```
use staging;

create table staging_oficina (

    id_oficina varchar(10) primary key,

    ciudad varchar(50),

    pais varchar(50),

    codigo_postal varchar(15),

    telefono varchar(20)
```

);

create table staging_empleado (

id_empleado int primary key,

nombre varchar(50),

apellido1 varchar(50),

id_oficina varchar(10),

id_jefe int,

puesto varchar(50),

foreign key (id_oficina) references staging_oficina(id_oficina)

-- nota: la fk a jefe se puede omitir en staging para evitar problemas de carga circular

);

create table staging_cliente (

id_cliente int primary key,

nombre_cliente varchar(100),

nombre_contacto varchar(50),

apellido_contacto varchar(50),

ciudad varchar(50),

pais varchar(50),

id_empleado_rep_ventas int,

limite_credito decimal(15,2),

foreign key (id_empleado_rep_ventas) references staging_empleado(id_empleado)

);

```
create table staging_producto (
```

```
    id_producto int primary key,
```

```
    nombre varchar(100),
```

```
    gama varchar(50),
```

```
    dimensiones varchar(50),
```

```
    proveedor varchar(100),
```

```
    precio_venta decimal(15,2)
```

```
);
```

```
create table staging_pedido (
```

```
    id_pedido int primary key,
```

```
    fecha_pedido date,
```

```
    fecha_entrega date,
```

```
    estado enum('pendiente','entregado','rechazado'),
```

```
    id_cliente int,
```

```
    foreign key (id_cliente) references staging_cliente(id_cliente)
```

```
);
```

```
create table staging_detalle_pedido (
```

```
    id_pedido int,
```

```
    id_producto int,
```

```
    cantidad int,
```

```
    precio_unidad decimal(15,2),
```

```
    primary key (id_pedido, id_producto),
```

```
    foreign key (id_pedido) references staging_pedido(id_pedido),
```

```
    foreign key (id_producto) references staging_producto(id_producto)
```

```
);

create table staging_pago (

    id_pago int auto_increment primary key,

    id_cliente int,

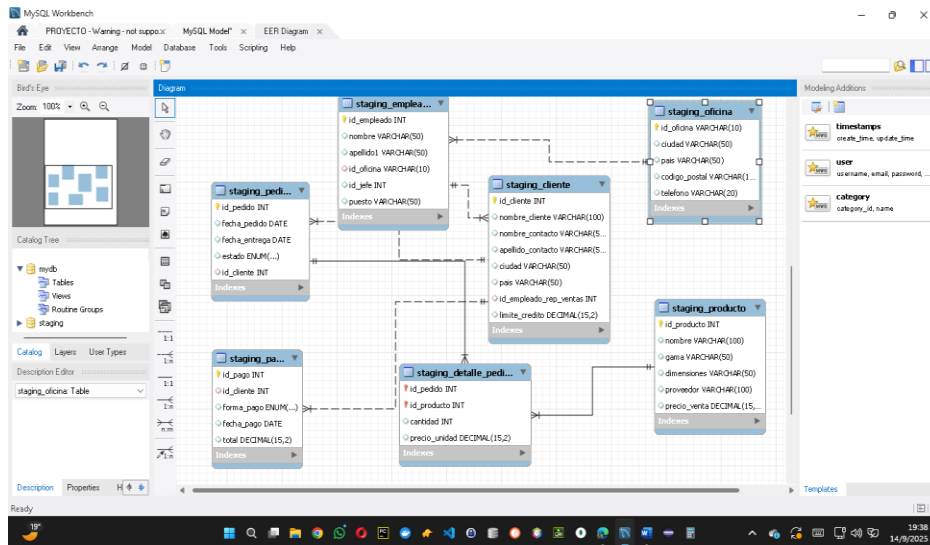
    forma_pago enum('transferencia','cheque','tarjeta'),

    fecha_pago date,

    total decimal(15,2),

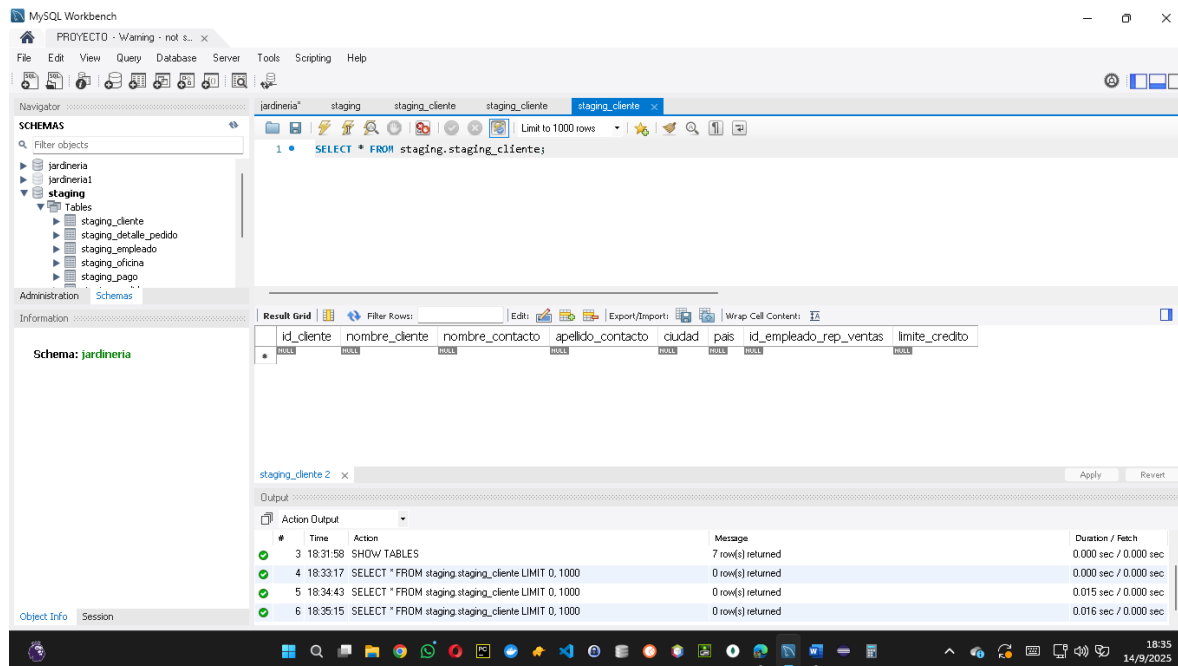
    foreign key (id_cliente) references staging_cliente(id_cliente)

);
```



realizar carga de datos desde bd jardinería a bd staging

tabla staging oficina vacía



script carga datos

insert into staging_oficina (id_oficina, ciudad, pais, codigo_postal, telefono)

select id_oficina, ciudad, pais, codigo_postal, telefono

from jardineria.oficina;

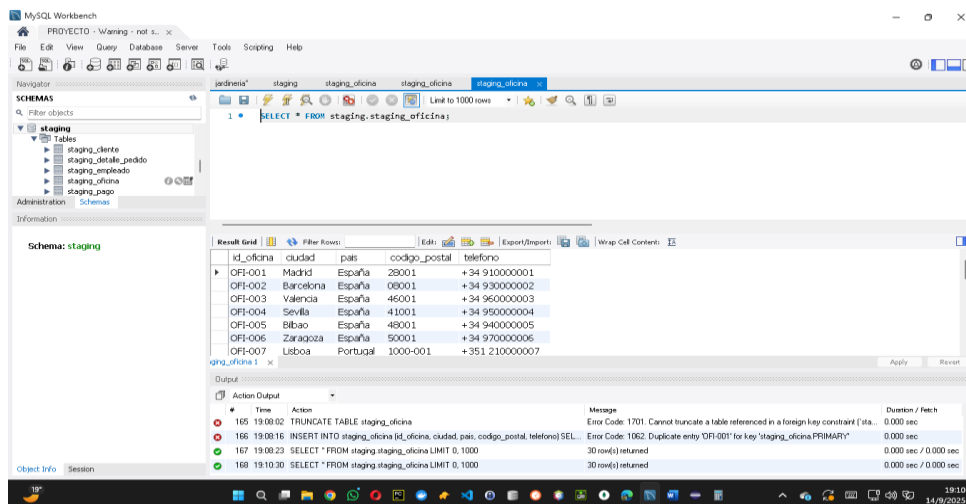
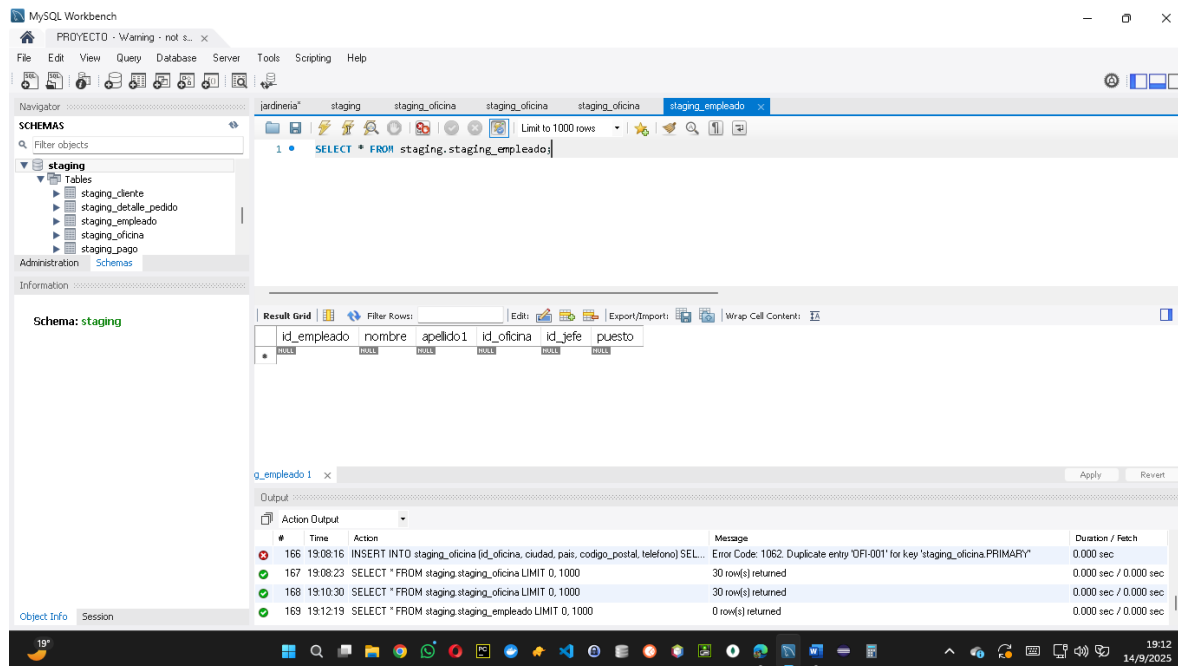


tabla staging empleado vacía



script carga datos

insert into staging_empleado (id_empleado, nombre, apellido1, puesto, id_jefe, id_oficina)

select id_empleado, nombre, apellido1, puesto, id_jefe, id_oficina
from jardineria.empleado;

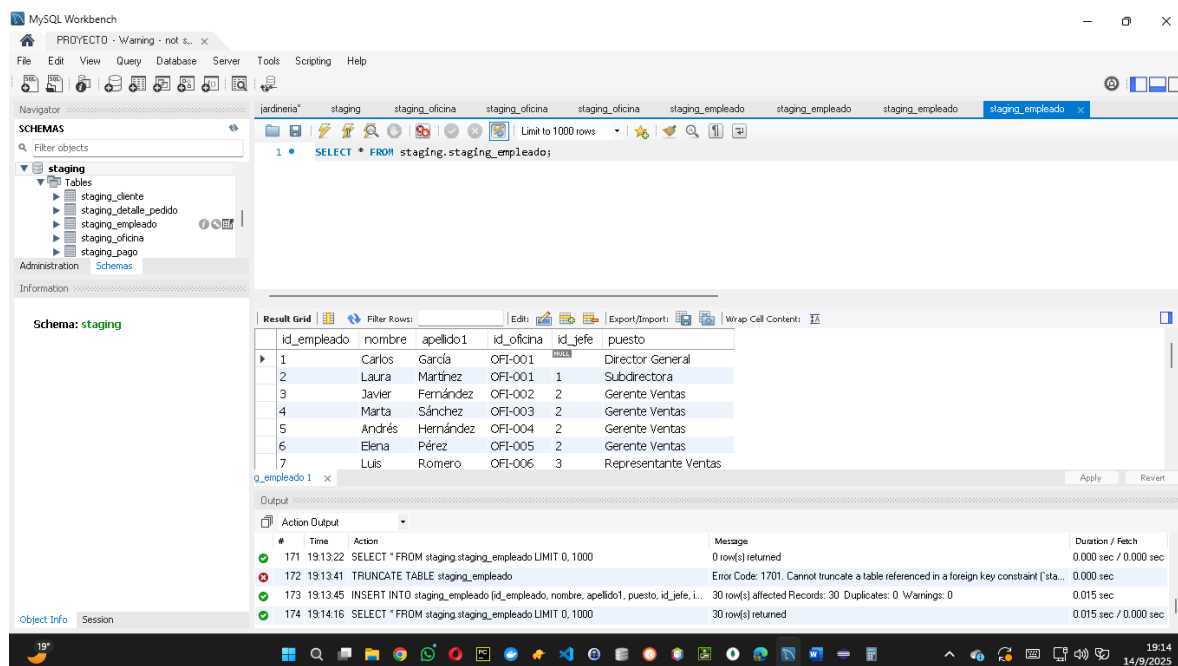
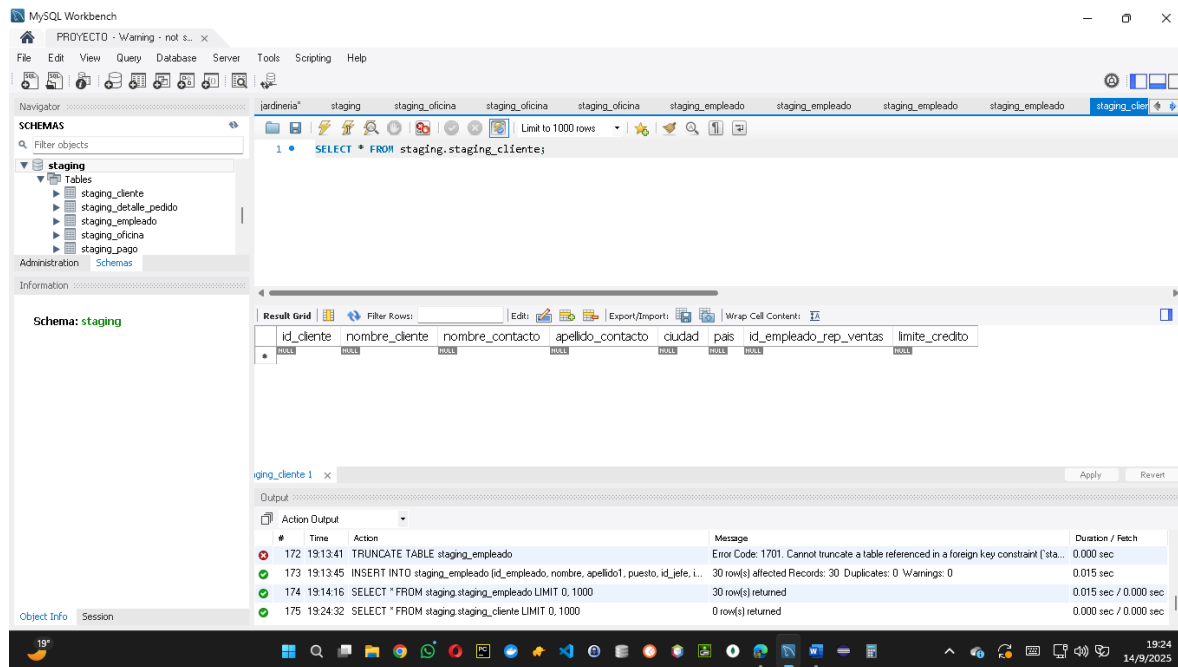


tabla staging cliente vacía



script carga datos

```
insert into staging_cliente (id_cliente, nombre_cliente, nombre_contacto,
apellido_contacto, ciudad, pais, id_empleado_rep_ventas, limite_credito)
```

```
select id_cliente, nombre_cliente, nombre_contacto, apellido_contacto, ciudad, pais,
id_empleado_rep_ventas, limite_credito
```

```
from jardineria.cliente;
```

Schema: staging

id_cliente	nombre_cliente	nombre_contacto	apellido_contacto	ciudad	pais	id_empleado_rep_ventas	limite_credito
101	Jardinería Verde	Ana	Martínez	Madrid	España	7	50000.00
102	Flores del Sol	Juan	López	Barcelona	España	8	45000.00
103	Plantas y Más	Lucía	Pérez	Valencia	España	9	60000.00
104	Jardín Urbano	Marcos	Sánchez	Sevilla	España	10	40000.00
105	Green World	Clara	Gómez	Bilbao	España	11	55000.00
106	EcoGarden	David	Fernández	Zaragoza	España	12	70000.00
107	Paisajismo Total	Laura	Ruiz	Lisboa	Portugal	13	65000.00

Output:

#	Time	Action	Message	Duration / Fetch
174	19:14:16	SELECT * FROM staging.staging_empleado LIMIT 0, 1000	30 row(s) returned	0.015 sec / 0.000 sec
175	19:24:32	SELECT * FROM staging.staging_cliente LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
176	19:25:07	INSERT INTO staging_cliente (id_cliente, nombre_cliente, nombre_contacto, apellido_contacto, ciudad, pais, id_empleado_rep_ventas, limite_credito) VALUES (101, 'Jardinería Verde', 'Ana', 'Martínez', 'Madrid', 'España', 7, 50000.00), (102, 'Flores del Sol', 'Juan', 'López', 'Barcelona', 'España', 8, 45000.00), (103, 'Plantas y Más', 'Lucía', 'Pérez', 'Valencia', 'España', 9, 60000.00), (104, 'Jardín Urbano', 'Marcos', 'Sánchez', 'Sevilla', 'España', 10, 40000.00), (105, 'Green World', 'Clara', 'Gómez', 'Bilbao', 'España', 11, 55000.00), (106, 'EcoGarden', 'David', 'Fernández', 'Zaragoza', 'España', 12, 70000.00), (107, 'Paisajismo Total', 'Laura', 'Ruiz', 'Lisboa', 'Portugal', 13, 65000.00)	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.031 sec
177	19:25:35	SELECT * FROM staging.staging_cliente LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec

tabla staging producto vacía

Schema: staging

id_producto	nombre	gama	dimensiones	proveedor	precio_venta

Output:

#	Time	Action	Message	Duration / Fetch
175	19:24:32	SELECT * FROM staging.staging_cliente LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
176	19:25:07	INSERT INTO staging_cliente (id_cliente, nombre_cliente, nombre_contacto, apellido_contacto, ciudad, pais, id_empleado_rep_ventas, limite_credito) VALUES (101, 'Jardinería Verde', 'Ana', 'Martínez', 'Madrid', 'España', 7, 50000.00), (102, 'Flores del Sol', 'Juan', 'López', 'Barcelona', 'España', 8, 45000.00), (103, 'Plantas y Más', 'Lucía', 'Pérez', 'Valencia', 'España', 9, 60000.00), (104, 'Jardín Urbano', 'Marcos', 'Sánchez', 'Sevilla', 'España', 10, 40000.00), (105, 'Green World', 'Clara', 'Gómez', 'Bilbao', 'España', 11, 55000.00), (106, 'EcoGarden', 'David', 'Fernández', 'Zaragoza', 'España', 12, 70000.00), (107, 'Paisajismo Total', 'Laura', 'Ruiz', 'Lisboa', 'Portugal', 13, 65000.00)	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.031 sec
177	19:25:35	SELECT * FROM staging.staging_cliente LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
178	19:27:08	SELECT * FROM staging.staging_producto LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

script carga datos

insert into staging_producto (id_producto, nombre, gama, dimensiones, proveedor, precio_venta)

select id_producto, nombre, gama, dimensiones, proveedor, precio_venta

from jardineria.producto;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'staging_pago', 'staging_pedido', 'staging_producto', 'Views', 'Stored Procedures', and 'Functions'. The 'Schema: staging' is selected. The main editor shows a query: `SELECT * FROM staging.staging_producto;`. The 'Result Grid' displays the following data:

id_producto	nombre	gama	dimensiones	proveedor	precio_venta
1001	Rosa Roja	Flores	30cm	Floricultura Madrid	2.50
1002	Tulipán Amarillo	Flores	25cm	Tulipanes BCN	3.00
1003	Orquídea Blanca	Orquídeas	40cm	Orquídeas Valencia	15.00
1004	Clavel Rojo	Flores	20cm	Claveles Sevilla	1.50
1005	Girasol	Flores	50cm	Campos Bilbao	2.80
1006	Cactus Mini	Suculentas	10cm	Green Lisboa	4.00
1007	Suculenta Aloe Vera	Suculentas	15cm	Portugal Verde	6.50

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
179	19:27:43	SELECT * FROM staging.staging_producto LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
180	19:27:43	INSERT INTO staging_producto (id_producto, nombre, gama, dimensiones, proveedor, precio_venta) VALUES (1001, 'Rosa Roja', 'Flores', '30cm', 'Floricultura Madrid', 2.50), (1002, 'Tulipán Amarillo', 'Flores', '25cm', 'Tulipanes BCN', 3.00), (1003, 'Orquídea Blanca', 'Orquídeas', '40cm', 'Orquídeas Valencia', 15.00), (1004, 'Clavel Rojo', 'Flores', '20cm', 'Claveles Sevilla', 1.50), (1005, 'Girasol', 'Flores', '50cm', 'Campos Bilbao', 2.80), (1006, 'Cactus Mini', 'Suculentas', '10cm', 'Green Lisboa', 4.00), (1007, 'Suculenta Aloe Vera', 'Suculentas', '15cm', 'Portugal Verde', 6.50)	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.016 sec
181	19:27:47	SELECT * FROM staging.staging_producto LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
182	19:27:58	SELECT * FROM staging.staging_producto LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec

tabla staging_pedido vacía

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'staging_pago', 'staging_pedido', 'staging_producto', 'Views', 'Stored Procedures', and 'Functions'. The 'Schema: staging' is selected. The main editor shows a query: `SELECT * FROM staging.staging_pedido;`. The 'Result Grid' is empty, indicating no data was returned. The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
180	19:27:43	INSERT INTO staging_producto (id_producto, nombre, gama, dimensiones, proveedor, precio_venta) VALUES (1001, 'Rosa Roja', 'Flores', '30cm', 'Floricultura Madrid', 2.50), (1002, 'Tulipán Amarillo', 'Flores', '25cm', 'Tulipanes BCN', 3.00), (1003, 'Orquídea Blanca', 'Orquídeas', '40cm', 'Orquídeas Valencia', 15.00), (1004, 'Clavel Rojo', 'Flores', '20cm', 'Claveles Sevilla', 1.50), (1005, 'Girasol', 'Flores', '50cm', 'Campos Bilbao', 2.80), (1006, 'Cactus Mini', 'Suculentas', '10cm', 'Green Lisboa', 4.00), (1007, 'Suculenta Aloe Vera', 'Suculentas', '15cm', 'Portugal Verde', 6.50)	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.016 sec
181	19:27:47	SELECT * FROM staging.staging_producto LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
182	19:27:58	SELECT * FROM staging.staging_producto LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
183	19:30:23	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

script carga datos

insert into staging_pedido (id_pedido, fecha_pedido, fecha_entrega, estado, id_cliente)

select id_pedido, fecha_pedido, fecha_entrega, estado, id_cliente

from jardineria.pedido;

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view containing 'staging_pago', 'staging_pedido', 'staging_producto', 'Views', 'Stored Procedures', and 'Functions'. The 'Schema: staging' is selected. The main query editor shows a query: `SELECT * FROM staging.staging_pedido;`. The 'Result Grid' displays the following data:

id_pedido	fecha_pedido	fecha_entrega	estado	id_cliente
2001	2025-09-01	2025-09-05	Pendiente	102
2002	2025-09-02	2025-09-05	Entregado	103
2003	2025-09-03	2025-09-07	Pendiente	104
2004	2025-09-04	2025-09-07	Entregado	105
2005	2025-09-05	2025-09-09	Pendiente	106
2006	2025-09-06	2025-09-09	Entregado	107
2007	2025-09-07		Pendiente	

The 'Output' panel shows the execution log:

#	Time	Action	Message	Duration / Fetch
183	19:30:23	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
184	19:31:35	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
185	19:31:44	INSERT INTO staging_pedido (id_pedido, fecha_pedido, fecha_entrega, estado, id...	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.031 sec
186	19:31:46	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
187	19:31:46	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec

tabla staging detalle pedido vacía

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view containing 'staging_detalle_pedido', 'staging_empleado', 'staging_oficina', 'staging_pago', 'staging_pedido', and 'staging_producto'. The 'Schema: staging' is selected. The main query editor shows a query: `SELECT * FROM staging.staging_detalle_pedido;`. The 'Result Grid' displays the following data:

id_pedido	id_producto	cantidad	precio_unidad
2001	1001	1000	1000

The 'Output' panel shows the execution log:

#	Time	Action	Message	Duration / Fetch
184	19:31:35	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
185	19:31:44	INSERT INTO staging_pedido (id_pedido, fecha_pedido, fecha_entrega, estado, id...	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.031 sec
186	19:31:46	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
187	19:32:19	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec

script carga datos

insert into staging_detalle_pedido (id_pedido, id_producto, cantidad, precio_unidad)

select id_pedido, id_producto, cantidad, precio_unidad

from jardineria.detalle_pedido;

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'staging' schema. The 'Query' editor contains the SQL query: `SELECT * FROM staging.staging_detalle_pedido;`. The 'Result Grid' displays the following data:

id_pedido	id_producto	cantidad	precio_unidad
2001	1001	10	2.50
2002	1002	15	3.00
2003	1003	5	15.00
2004	1004	20	1.50
2005	1005	12	2.80
2006	1006	8	4.00
2007	1007	7	6.50

The 'Output' pane shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
186	19:31:46	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
187	19:32:19	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
188	19:33:36	INSERT INTO staging_detalle_pedido (id_pedido, id_producto, cantidad, precio_uni...	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.016 sec
189	19:33:38	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
190	19:34:30	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

tabla staging pagos vacía

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left lists the 'staging' schema. The 'Query' editor contains the SQL query: `SELECT * FROM staging.staging_pago;`. The 'Result Grid' is empty, indicating no data was returned. The 'Output' pane shows the execution log with the following messages:

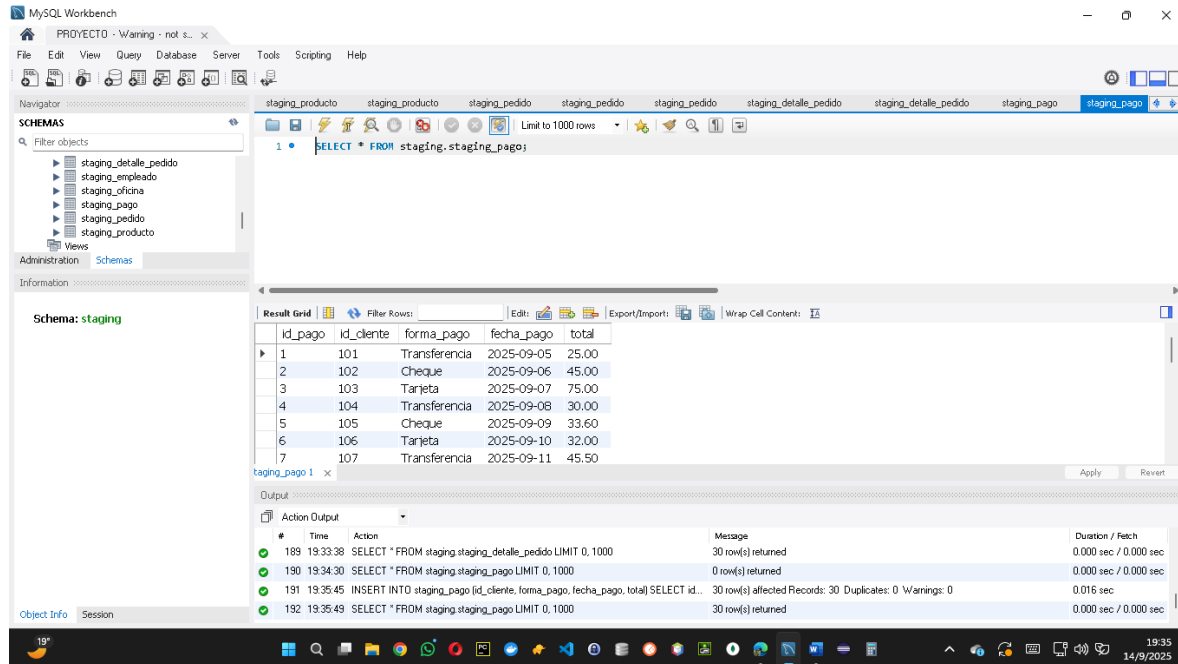
#	Time	Action	Message	Duration / Fetch
187	19:32:19	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	0 row(s) returned	0.016 sec / 0.000 sec
188	19:33:36	INSERT INTO staging_detalle_pedido (id_pedido, id_producto, cantidad, precio_uni...	30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0	0.016 sec
189	19:33:38	SELECT * FROM staging.staging_detalle_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
190	19:34:30	SELECT * FROM staging.staging_pago LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

script carga datos

insert into staging_pago (id_cliente, forma_pago, fecha_pago, total)

select id_cliente, forma_pago, fecha_pago, total

from jardineria.pago;



The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' panel with a tree view of databases including 'staging_detalle_pedido', 'staging_empleado', 'staging_oficina', 'staging_pago', 'staging_pedido', and 'staging_producto'. The 'staging_pago' database is selected. The main query editor shows a SQL query: `SELECT * FROM staging.staging_pago;`. The 'Result Grid' at the bottom displays the query results in a table format. The table has columns: 'id_pago', 'id_cliente', 'forma_pago', 'fecha_pago', and 'total'. The results show 7 rows of data. The 'Output' panel at the bottom shows the execution log with messages for the query execution.

#	id_pago	id_cliente	forma_pago	fecha_pago	total
1	101	101	Transferencia	2025-09-05	25.00
2	102	102	Cheque	2025-09-06	45.00
3	103	103	Tarjeta	2025-09-07	75.00
4	104	104	Transferencia	2025-09-08	30.00
5	105	105	Cheque	2025-09-09	33.60
6	106	106	Tarjeta	2025-09-10	32.00
7	107	107	Transferencia	2025-09-11	45.50

conteo de los datos en cada tabla bd staging

-- conteo en staging

select 'oficina' as tabla, count(*) as registros from staging_oficina

union all

select 'empleado', count(*) from staging_empleado

union all

select 'cliente', count(*) from staging_cliente

union all

select 'producto', count(*) from staging_producto

union all

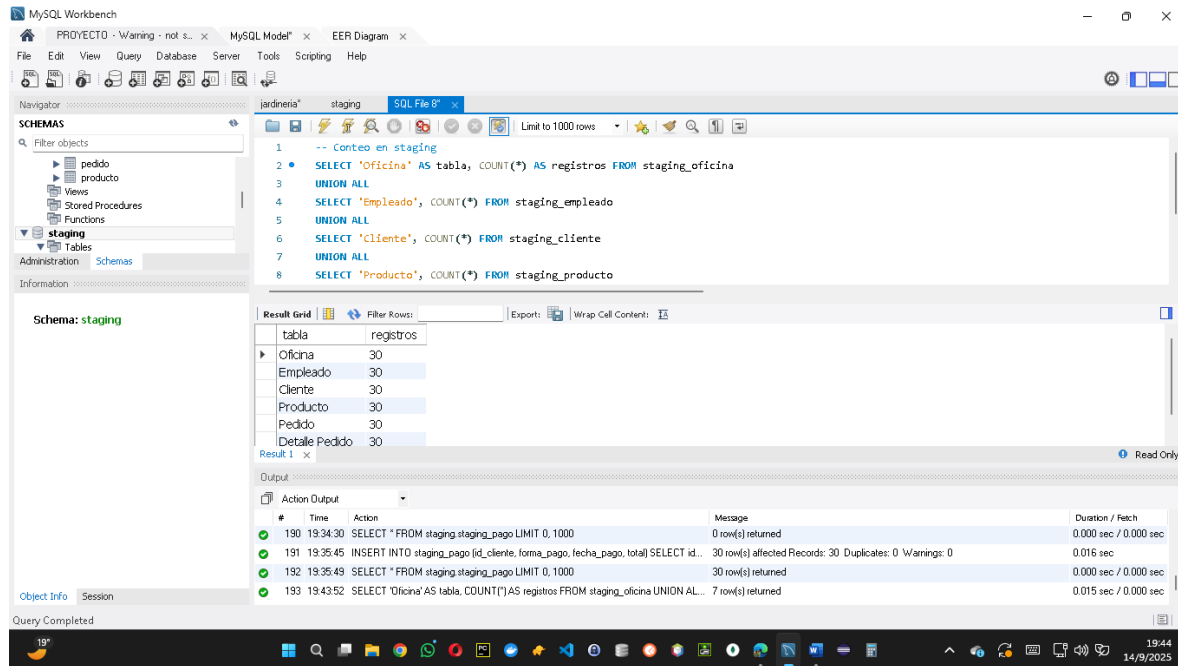
select 'pedido', count(*) from staging_pedido

union all


```
select 'detalle pedido', count(*) from staging_detalle_pedido
```

```
union all
```

```
select 'pago', count(*) from staging_pago;
```



script para validar datos cargados en bd staging

```
-- oficinas
```

```
select * from staging_oficina limit 5;
```

```
-- empleados
```

```
select * from staging_empleado limit 5;
```

```
-- clientes
```

```
select * from staging_cliente limit 5;
```

```
-- productos
```

```
select * from staging_producto limit 5;
```

```
-- pedidos
```

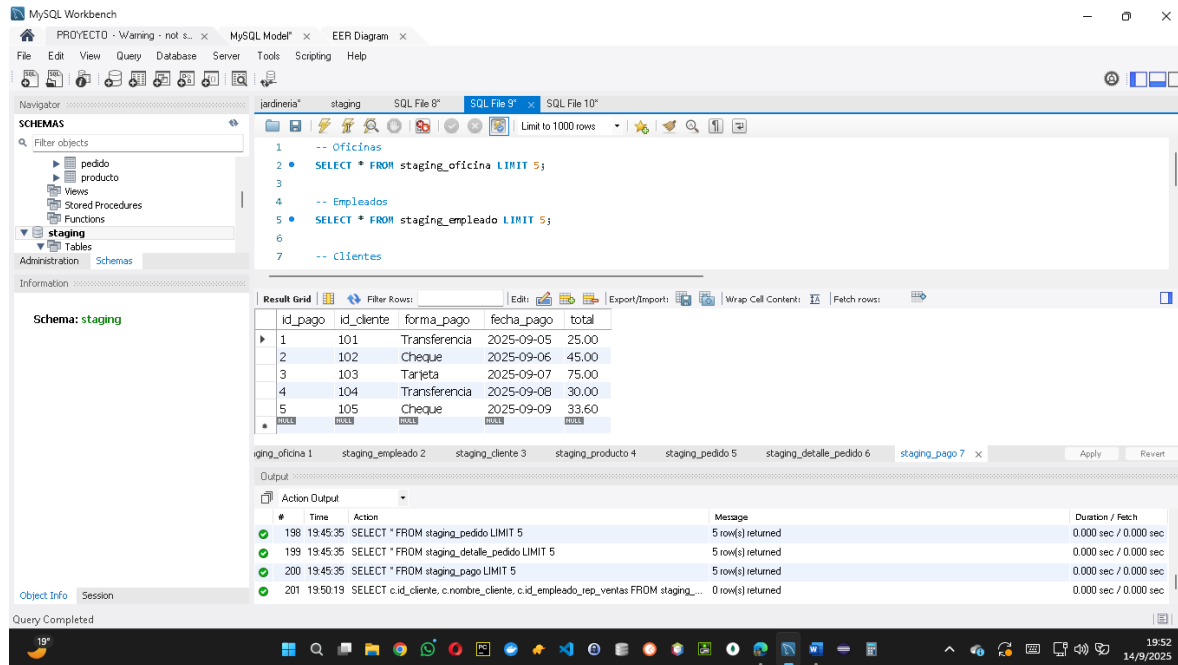
```
select * from staging_pedido limit 5;
```

-- detalles de pedidos

```
select * from staging_detalle_pedido limit 5;
```

-- pagos

```
select * from staging_pago limit 5;
```



MySQL Workbench interface showing a query execution. The query is:

```
-- Oficinas
SELECT * FROM staging_oficina LIMIT 5;

-- Empleados
SELECT * FROM staging_empleado LIMIT 5;

-- Clientes
```

The result grid displays the following data:

	id_pago	id_cliente	forma_pago	fecha_pago	total
1	101	101	Transferencia	2025-09-05	25.00
2	102	102	Cheque	2025-09-06	45.00
3	103	103	Tarjeta	2025-09-07	75.00
4	104	104	Transferencia	2025-09-08	30.00
5	105	105	Cheque	2025-09-09	33.60

The action output shows the following results:

#	Time	Action	Message	Duration / Fetch
198	19:45:35	SELECT * FROM staging_pedido LIMIT 5	5 row(s) returned	0.000 sec / 0.000 sec
199	19:45:35	SELECT * FROM staging_detalle_pedido LIMIT 5	5 row(s) returned	0.000 sec / 0.000 sec
200	19:45:35	SELECT * FROM staging_pago LIMIT 5	5 row(s) returned	0.000 sec / 0.000 sec
201	19:50:19	SELECT c.id_cliente, c.nombre_cliente, c.id_empleado_rep_ventas FROM staging...	0 row(s) returned	0.000 sec / 0.000 sec

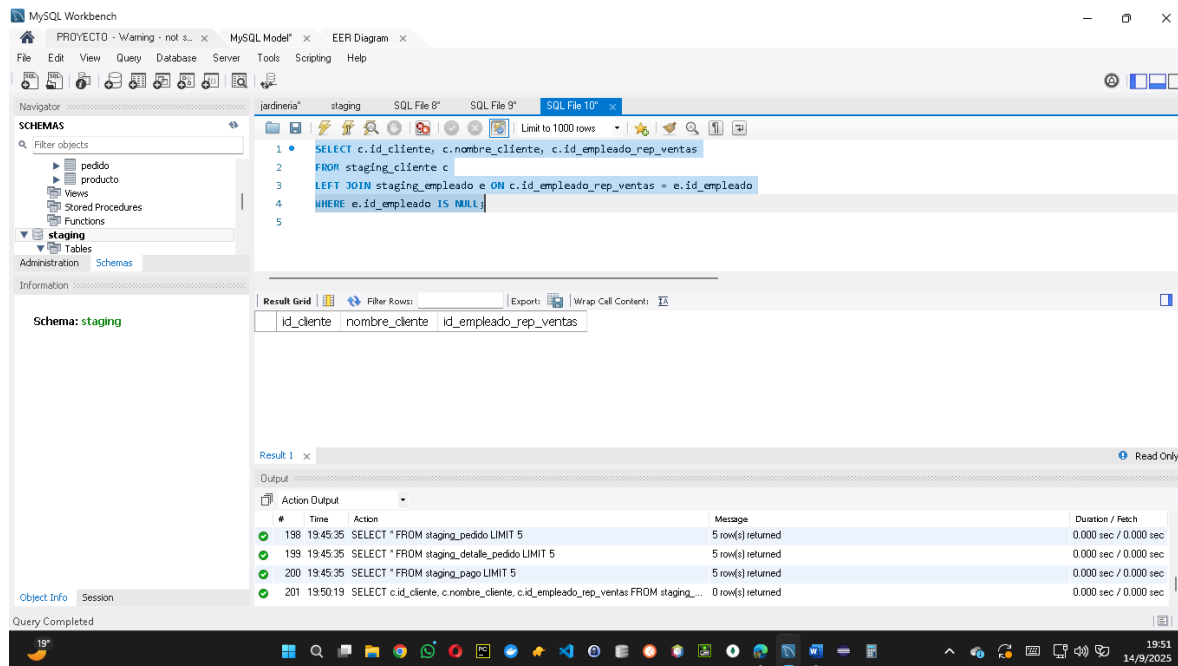
validar integridad de relaciones

```
select c.id_cliente, c.nombre_cliente, c.id_empleado_rep_ventas
```

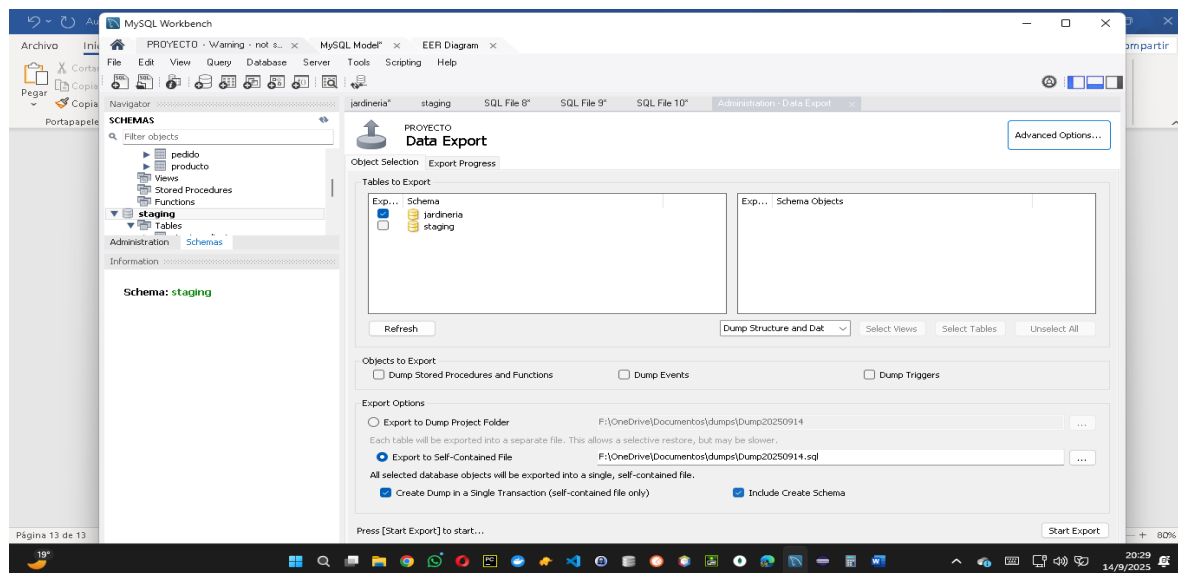
```
from staging_cliente c
```

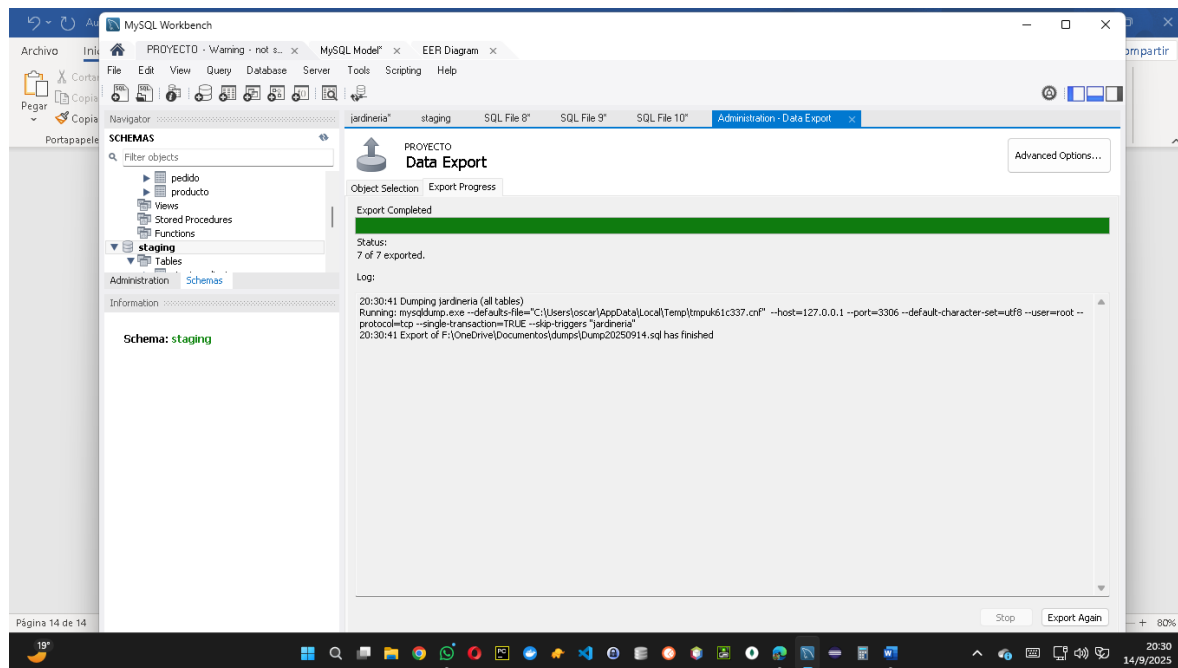
```
left join staging_empleado e on c.id_empleado_rep_ventas = e.id_empleado
```

```
where e.id_empleado is null;
```

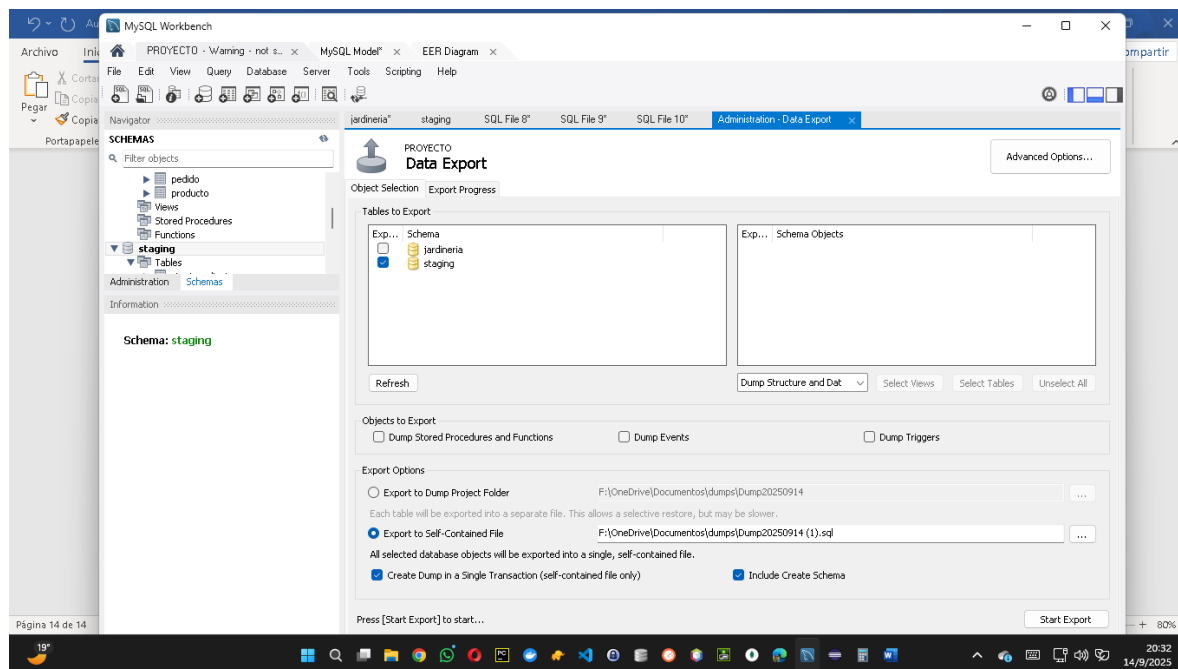


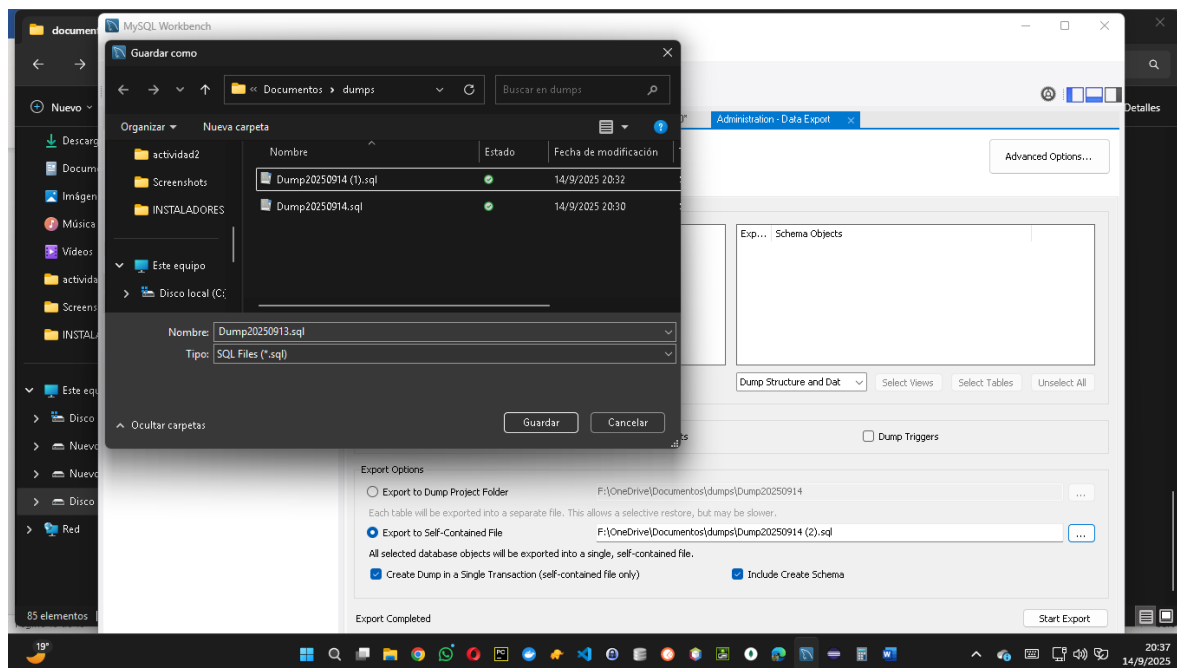
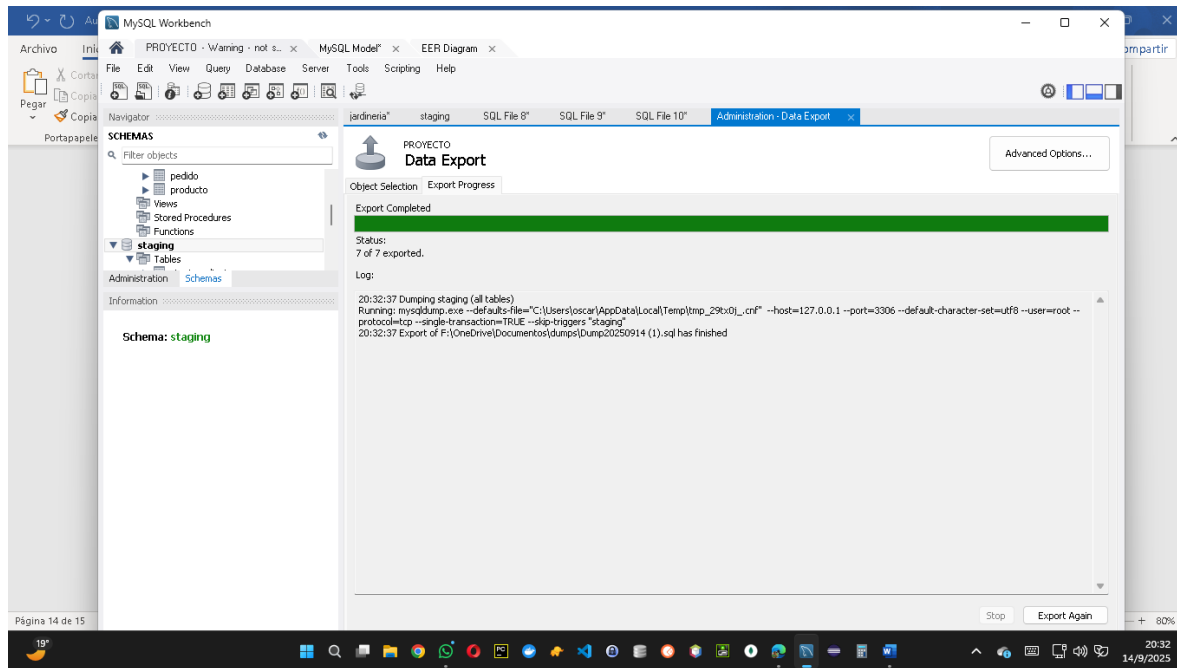
Bk de BD jardineria





Bk de BD Staging





Conclusiones

La construcción de la base de datos *Staging* a partir de Jardinería nos permitió entender la importancia de contar con un espacio intermedio en los procesos de integración y análisis de datos. Durante el trabajo pudimos identificar que la base original, aunque muy útil para las operaciones transaccionales, no estaba preparada para un uso analítico por la cantidad de tablas, atributos secundarios y relaciones complejas que maneja.

Con el diseño de Staging logramos simplificar la estructura, seleccionando únicamente los campos más relevantes de cada tabla, lo que facilita el manejo de la información y garantiza una mayor consistencia al momento de realizar consultas o trasladar los datos a un futuro data warehouse.

Además, este ejercicio nos ayudó a aplicar los conceptos de ETL de manera práctica, viendo cómo la extracción, transformación y carga de datos deben planearse de forma cuidadosa para evitar errores y asegurar que la información llegue limpia y organizada.

En conclusión, el proyecto de Staging no solo resolvió las dificultades de la base Jardinería para fines analíticos, sino que también nos permitió comprender el valor de preparar los datos antes de usarlos en sistemas más complejos de análisis y toma de decisiones.

Bibliografía

Manso Millán, J. A. (2021). Retos de la migración de datos hacia un Data Warehouse. Obtenido de https://oa.upm.es/67979/1/TFG_JUAN_ANTONIO_MANSO_MILLAN.pdf