Institución Universitaria Digital de Antioquia

Evidencia de aprendizaje 3. Proceso de transformación de datos y carga en el datamart final

Integrantes:

Anderson Gaviria Bedoya

Robert Andrés Castillo Gaviria

Oscar Javier García García

María Fernanda Vásquez Montiel

Matrícula: PREICA2502B010064

Unidad 2:

Datamart

Nombre Tutor: Antonio Jesús Valderrama

Fecha de elaboración: 28 de septiembre de 2025

INTRODUCCIÓN

En la actualidad, las organizaciones generan grandes volúmenes de información a partir de sus procesos comerciales, transacciones y operaciones diarias. sin embargo, estos datos suelen estar dispersos en diferentes sistemas y bases de datos, lo que dificulta su integración y análisis para la toma de decisiones estratégicas.

un **DATA MART** permite organizar y consolidar la información en estructuras optimizadas, enfocadas en áreas específicas del negocio, facilitando el análisis de indicadores clave de desempeño. para ello, se emplea un proceso **ETL** (extracción, transformación y carga) que garantiza que los datos pasen desde la fuente original hasta el DATA MART de manera confiable y estandarizada.

en este proyecto se desarrolla un proceso **ETL** para el área de ventas de una empresa de jardinería, con el fin de identificar tendencias, productos más vendidos, desempeño de los empleados y comportamiento de los clientes.

Objetivos

Objetivo General

Desarrollar un proceso **ETL** que permita transformar los datos de la base de datos transaccional de la empresa de jardinería hacia un **DATA MART** de ventas, con el propósito de facilitar el análisis de información y apoyar la toma de decisiones.

Objetivos Específicos

1. Identificar y comprender la estructura de la base de datos original de la empresa.

2. Diseñar un esquema en staging que permita la carga temporal y limpieza de datos.

3. Implementar un modelo dimensional (esquema estrella) compuesto por tablas de hechos y dimensiones.

4. Desarrollar los procesos de carga hacia el **DATA MART.**

5. Realizar consultas analíticas que permitan obtener indicadores como el producto más vendido, los clientes con mayores compras y el desempeño de los empleados.

Planteamiento del Problema

La empresa de jardinería cuenta con una base de datos transaccional que registra pedidos, clientes, productos y empleados. No obstante, estos datos no están organizados para un análisis eficiente, ya que el diseño relacional está orientado a las operaciones diarias y no al análisis estratégico.

Debido a esto, la gerencia enfrenta dificultades para responder preguntas clave como:

- ¿Cuál es el producto más vendido en un año determinado?
- ¿Qué clientes representan mayor valor para la empresa?
- ¿Cuál es el desempeño de los representantes de ventas?

La falta de un sistema que transforme los datos en información analítica limita la capacidad de la empresa para identificar tendencias, optimizar recursos y mejorar sus estrategias comerciales.

Análisis del Problema

- La base de datos transaccional no está diseñada para realizar análisis multidimensionales ni consultas agregadas.
- El gran volumen de registros y su estructura normalizada dificultan las consultas analíticas.
- Se carece de un mecanismo de integración que permita depurar y estandarizar los datos.
- La información no está centralizada en un modelo que favorezca la toma de decisiones.

Propuesta de Solución

La solución planteada consiste en implementar un **DATA MART** de ventas mediante un proceso **ETL**, siguiendo estas fases:

1. Extracción: Se extraen los datos relevantes desde la base de datos original hacia un entorno de staging.
2. Transformación: Se limpian, depuran y estandarizan los datos, garantizando consistencia en fechas, claves foráneas y métricas de negocio.
3. Carga: Se construye un modelo dimensional en un esquema estrella con una tabla de hechos (fact_ventas) y varias tablas de dimensiones (tiempo, cliente, producto, empleado).
4. Consultas analíticas: Se realizan querys para obtener información clave, como el producto más vendido, los mejores clientes y el desempeño de empleados.

De esta manera, la empresa contará con una herramienta que facilite el análisis de información, mejore la toma de decisiones y fortalezca sus estrategias comerciales.

Propuesta de Solución

La solución planteada consiste en implementar un **DATA MART** de ventas mediante un proceso **ETL,** siguiendo estas fases:

1. Extracción: Se extraen los datos relevantes desde la base de datos original hacia un entorno de staging.
2. Transformación: Se limpian, depuran y estandarizan los datos, garantizando consistencia en fechas, claves foráneas y métricas de negocio.
3. Carga: Se construye un modelo dimensional en un esquema estrella con una tabla de hechos (fact_ventas) y varias tablas de dimensiones (tiempo, cliente, producto, empleado).
4. Consultas analíticas: Se realizan queries para obtener información clave, como el producto más vendido, los mejores clientes y el desempeño de empleados.

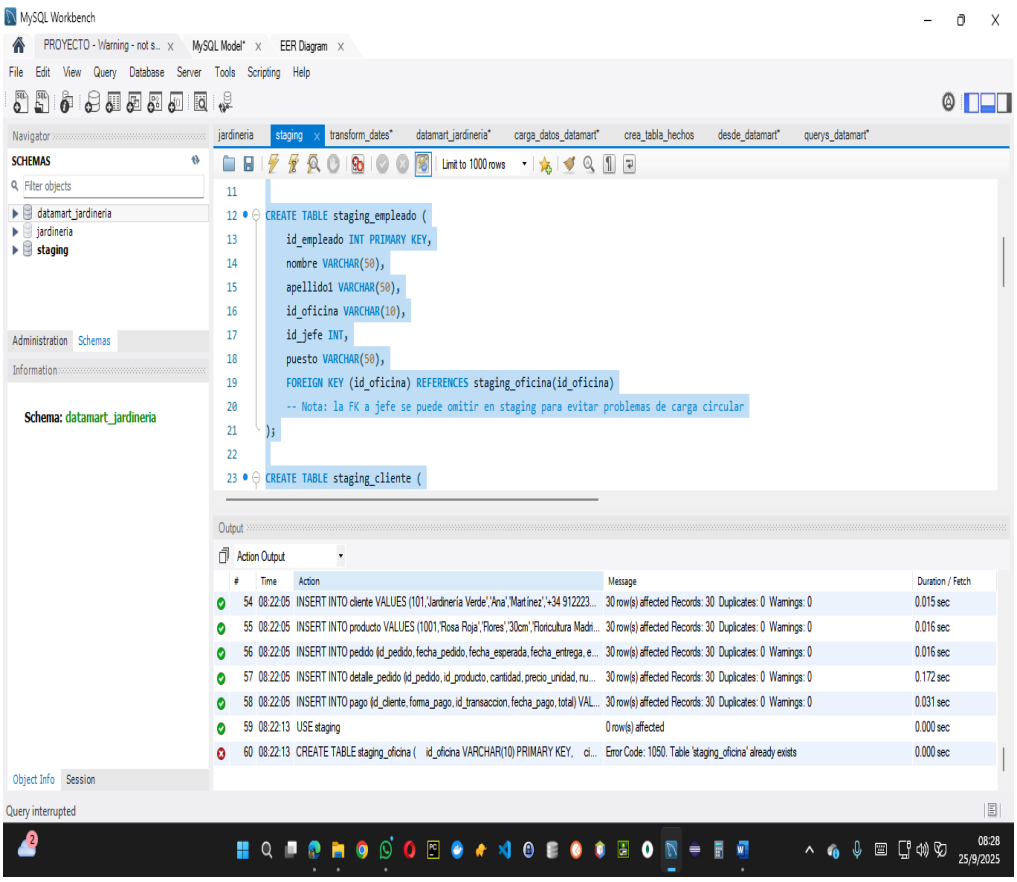
De esta manera, la empresa contará con una herramienta que facilite el análisis de información, mejore la toma de decisiones y fortalezca sus estrategias comerciales.

1.SCRIPT BASE DATOS STAGING PARA VERIFICACIÓN

```
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,
```

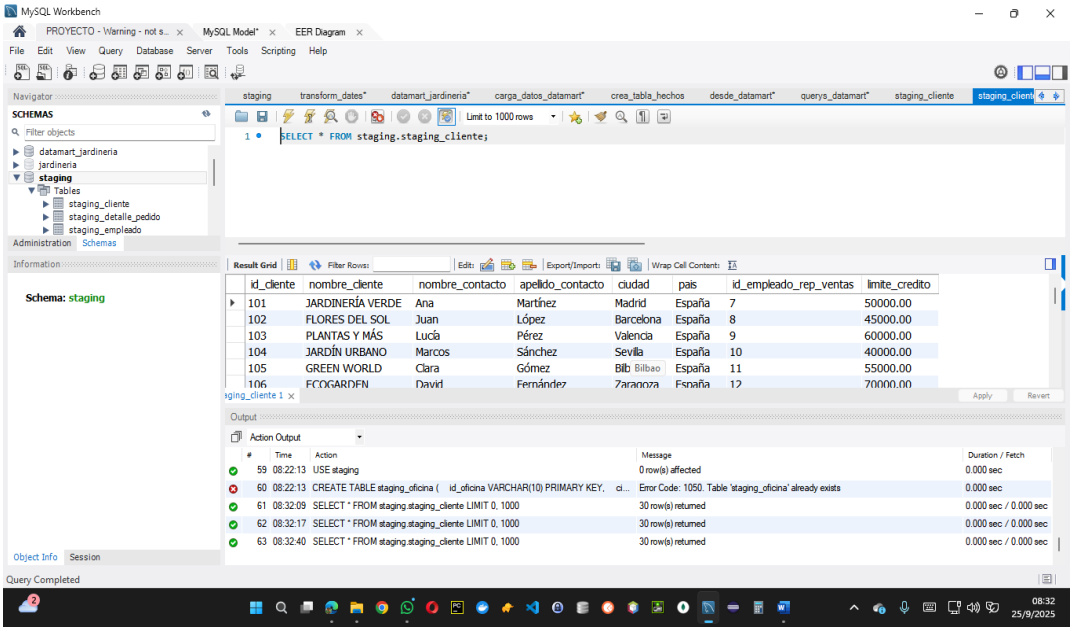
```
    estado ENUM('Pendiente','Entregado','Rechazado'),
    id_cliente INT,
    FOREIGN KEY (id_cliente) REFERENCES staging_cliente(id_cliente)
);

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)
);
```


MySQL Workbench
PROYECTO - Warning - not s...
MySQL Model*
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
datamart_jardineria
jardineria
staging
Administration Schemas
Information
Schema: datamart_jardineria
Object Info Session
jardineria staging transform_dates* datamart_jardineria* carga_datos_datamart* crea_tabla_hechos desde_datamart* querys_datamart*
Limit to 1000 rows
11
12 CREATE TABLE staging_empleado (
13 id_empleado INT PRIMARY KEY,
14 nombre VARCHAR(50),
15 apellido1 VARCHAR(50),
16 id_oficina VARCHAR(10),
17 id_jefe INT,
18 puesto VARCHAR(50),
19 FOREIGN KEY (id_oficina) REFERENCES staging_oficina(id_oficina)
20 -- Nota: la FK a jefe se puede omitir en staging para evitar problemas de carga circular
21);
22
23 CREATE TABLE staging_cliente (
Output
Action Output
Time Action Message Duration / Fetch
54 08:22:05 INSERT INTO cliente VALUES (101,'Jardinería Verde','Ana','Martínez','+34 912223... 30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0 0.015 sec
55 08:22:05 INSERT INTO producto VALUES (1001,'Rosa Roja','Flores','30cm','Floricultura Madri... 30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0 0.016 sec
56 08:22:05 INSERT INTO pedido (id_pedido, fecha_pedido, fecha_esperada, fecha_entrega, e... 30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0 0.016 sec
57 08:22:05 INSERT INTO detalle_pedido (id_pedido, id_producto, cantidad, precio_unidad, nu... 30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0 0.172 sec
58 08:22:05 INSERT INTO pago (id_cliente, forma_pago, id_transaccion, fecha_pago, total) VAL... 30 row(s) affected Records: 30 Duplicates: 0 Warnings: 0 0.031 sec
59 08:22:13 USE staging 0 row(s) affected 0.000 sec
60 08:22:13 CREATE TABLE staging_oficina (id_oficina VARCHAR(10) PRIMARY KEY, ci... Error Code: 1050. Table 'staging_oficina' already exists 0.000 sec
Query interrupted
08:28
25/9/2025

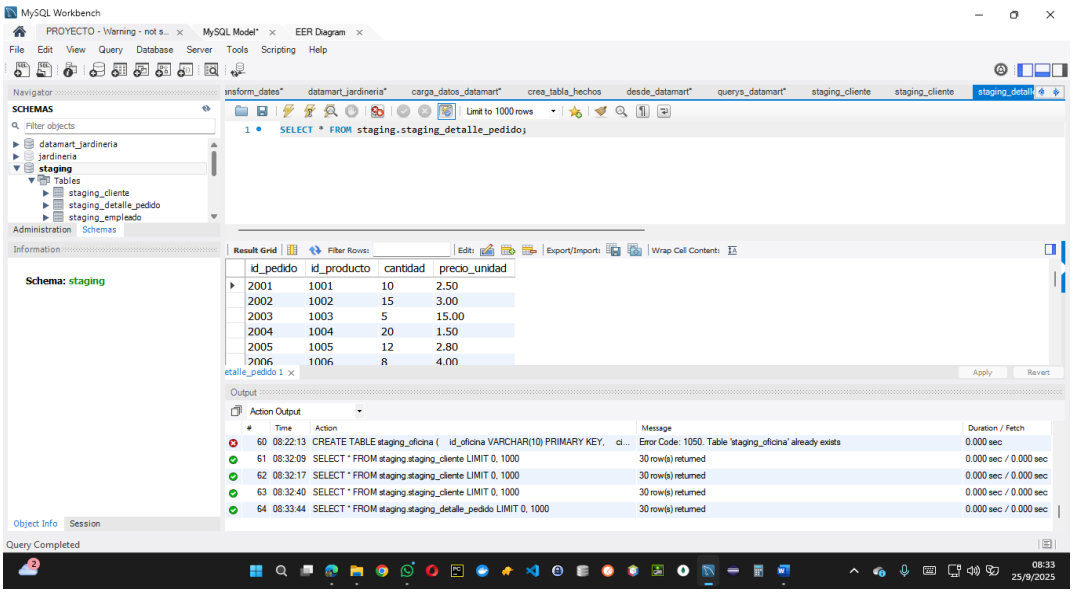
2.EXTRACCIÓN DE DATOS DESDE LA BASE DE DATOS ORIGEN HACIA LA BASE DE DATOS DE STAGING:

A) CARGA DE BASE DE DATOS ORIGINAL A STAGING VERIFICACIÓN DE TABLAS POBLADAS

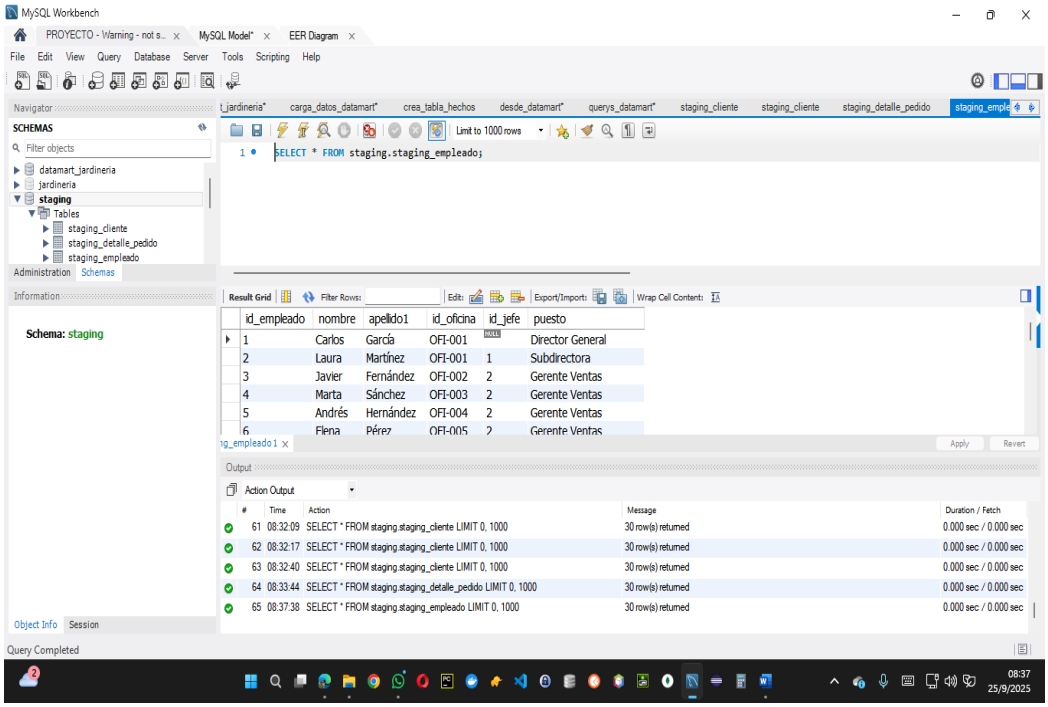
Detalle cliente



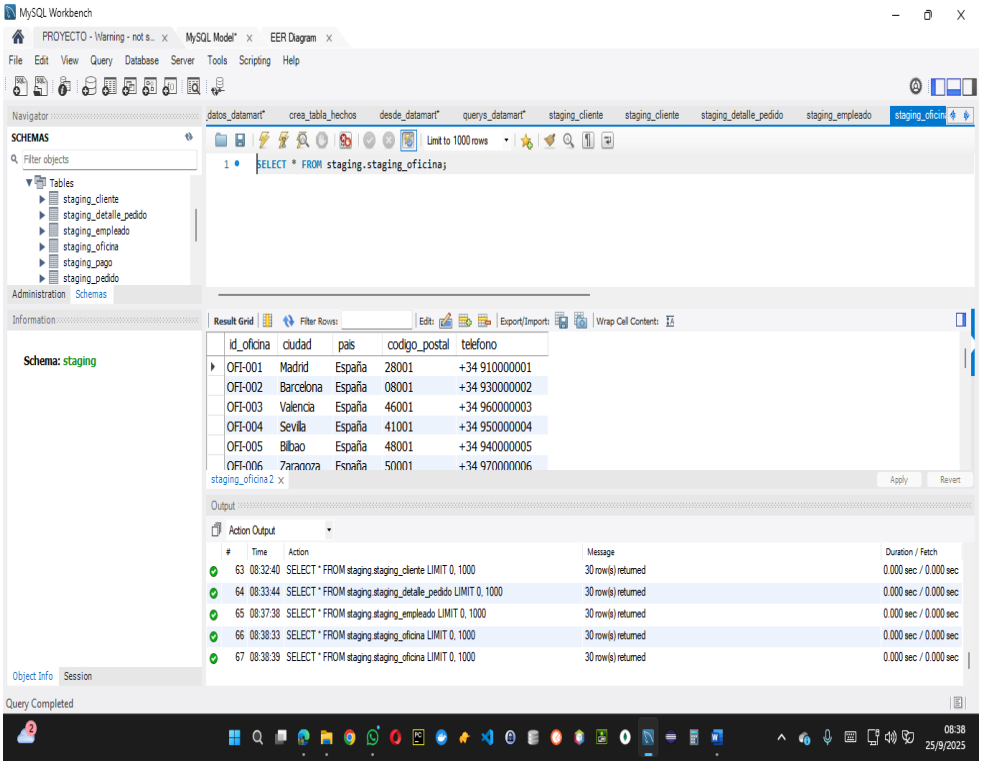
detalle pedido



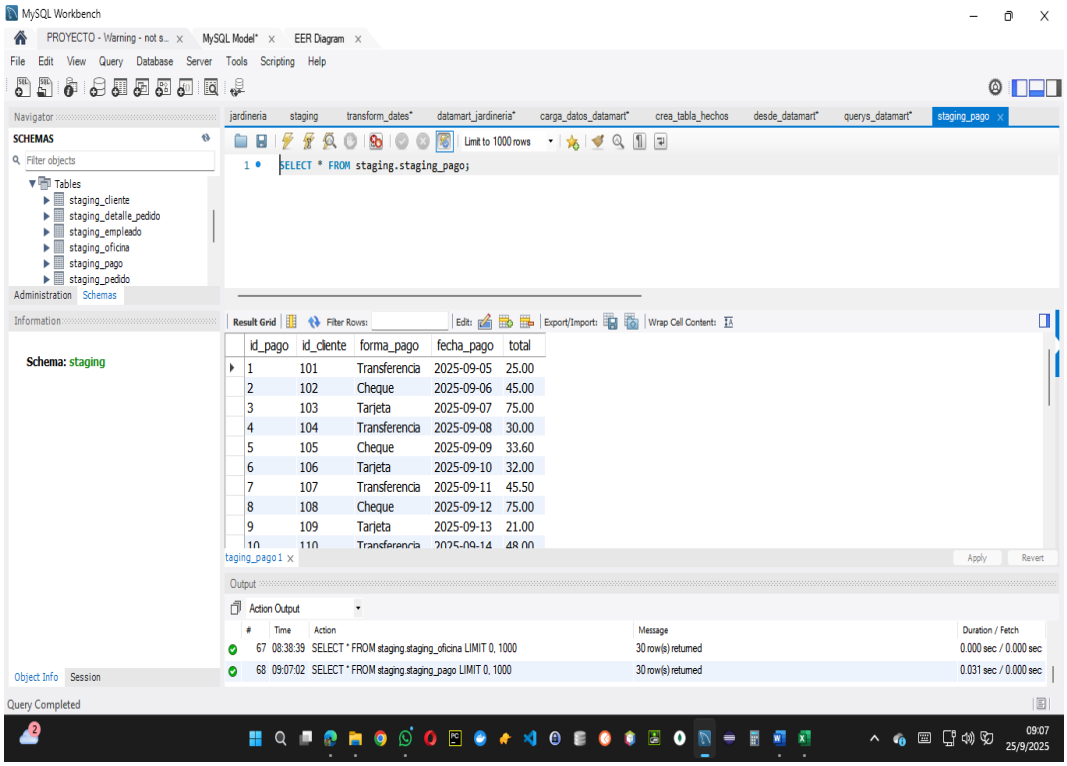
Detalle empleado



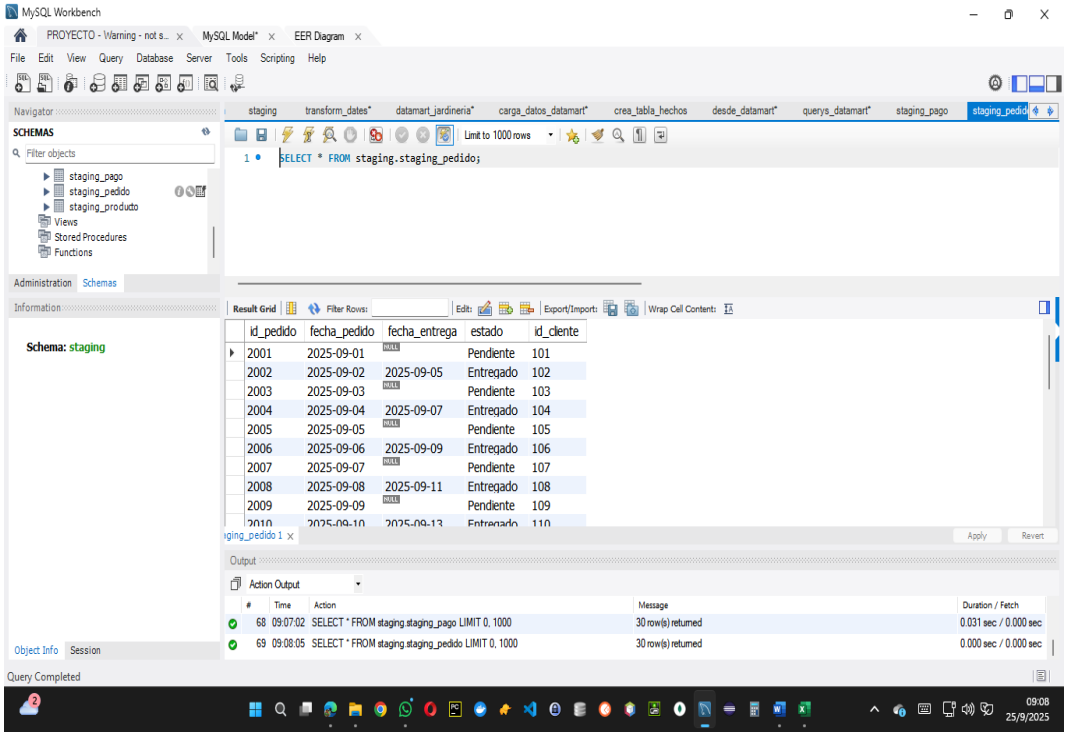
Detalle oficina



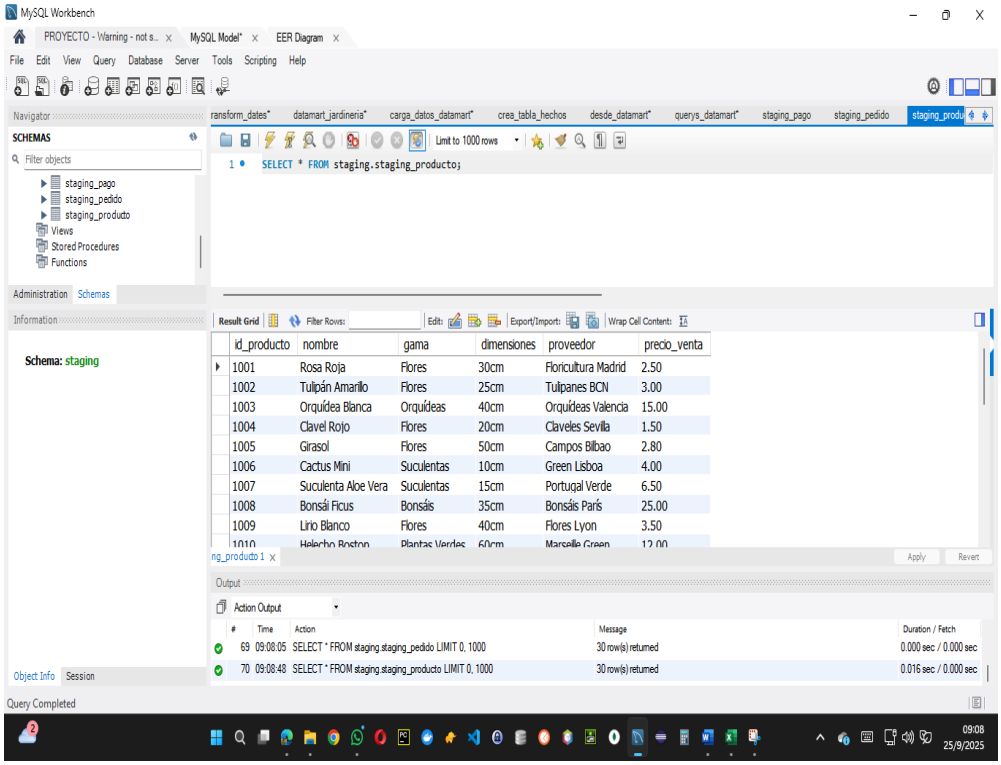
Detalle pago



Detalle pedido



Detalle producto



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
1008	Bonsái Ficus	Bonsáis	35cm	Bonsáis París	25.00
1009	Lirio Blanco	Flores	40cm	Flores Lyon	3.50

#	Time	Action	Message	Duration / Fetch
69	09:08:05	SELECT * FROM staging.staging_pedido LIMIT 0, 1000	30 row(s) returned	0.000 sec / 0.000 sec
70	09:08:48	SELECT * FROM staging.staging_producto LIMIT 0, 1000	30 row(s) returned	0.016 sec / 0.000 sec

A) SCRIPTS PARA POBLAR BD STAGING

insert into staging_oficina (id_oficina, ciudad, pais, codigo_postal, telefono) select id_oficina, ciudad, pais, codigo_postal, telefono from jardineria.oficina;

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;

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

nombre_cliente, nombre_contacto, apellido_contacto, ciudad, pais, id_empleado_rep_ventas, limite_credito from jardineria.cliente;

insert into staging_producto (id_producto, nombre, gama, dimensiones, proveedor, precio_venta) select id_producto, nombre, gama, dimensiones, proveedor, precio_venta from jardineria.producto;

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;

insert into staging_detalle_pedido (id_pedido, id_producto, cantidad, precio_unidad) select id_pedido, id_producto, cantidad, precio_unidad from jardineria.detalle_pedido;

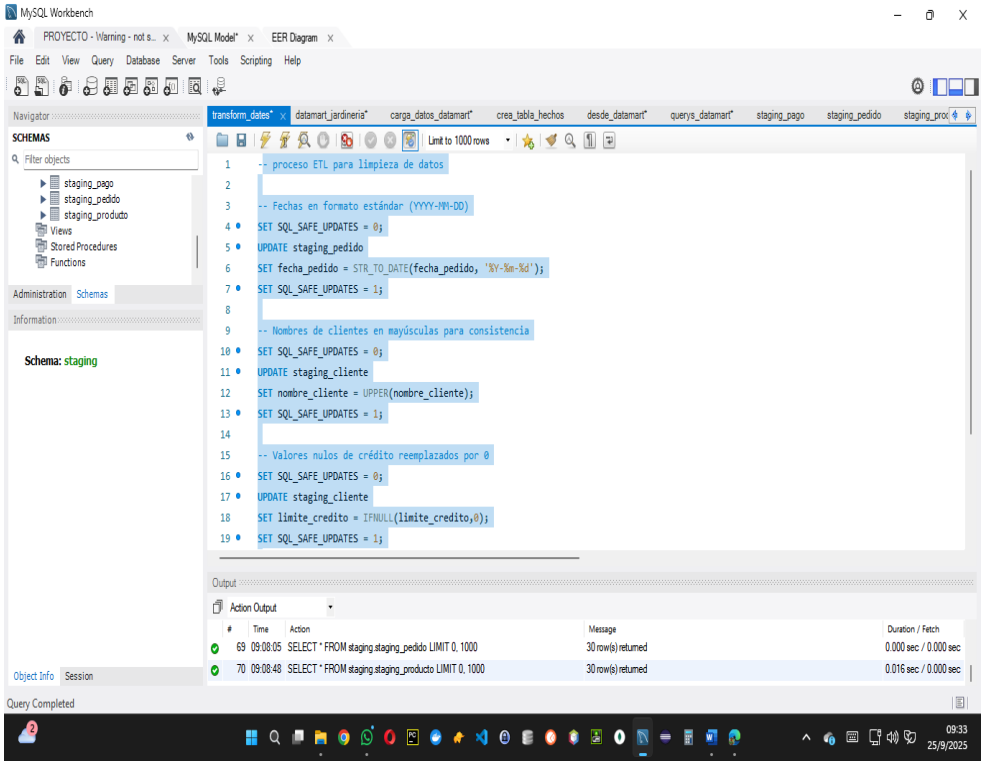
insert into staging_pago (id_cliente, forma_pago, fecha_pago, total) select id_cliente, forma_pago, fecha_pago, total from jardineria.pago;

B) 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;

3. TRANSFORMACIÓN DE DATOS SEGÚN LAS NECESIDADES ANALÍTICAS:

QUERYS PARA ANÁLISIS Y ORGANIZACIÓN DE DATOS TRANSFORMARLOS



- PROCESO ETL PARA LIMPIEZA DE DATOS

-- Fechas en formato estándar (YYYY-MM-DD)

SET SQL_SAFE_UPDATES = 0;

UPDATE staging_pedido

SET fecha_pedido = STR_TO_DATE(fecha_pedido, '%Y-%m-%d');

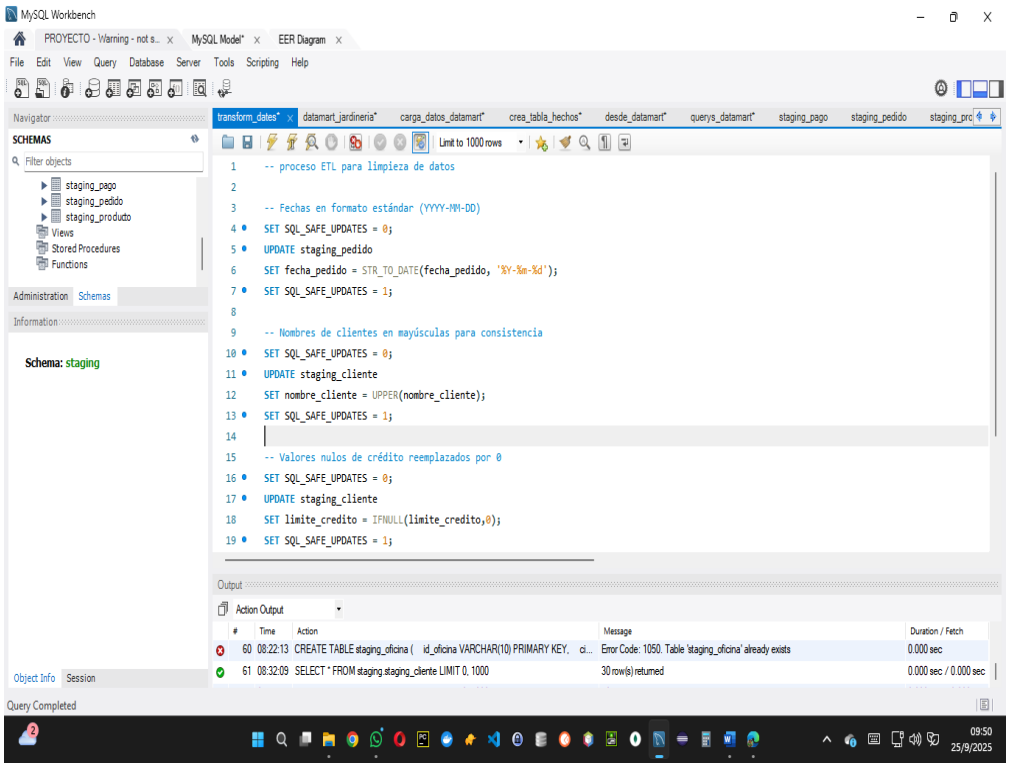
SET SQL_SAFE_UPDATES = 1;

-- Nombres de clientes en mayúsculas para consistencia

SET SQL_SAFE_UPDATES = 0;

UPDATE staging_cliente

```
SET nombre_cliente = UPPER(nombre_cliente);

SET SQL_SAFE_UPDATES = 1;


-- Valores nulos de crédito reemplazados por 0

SET SQL_SAFE_UPDATES = 0;

UPDATE staging_cliente

SET limite_credito = IFNULL(limite_credito,0);

SET SQL_SAFE_UPDATES = 1;
```

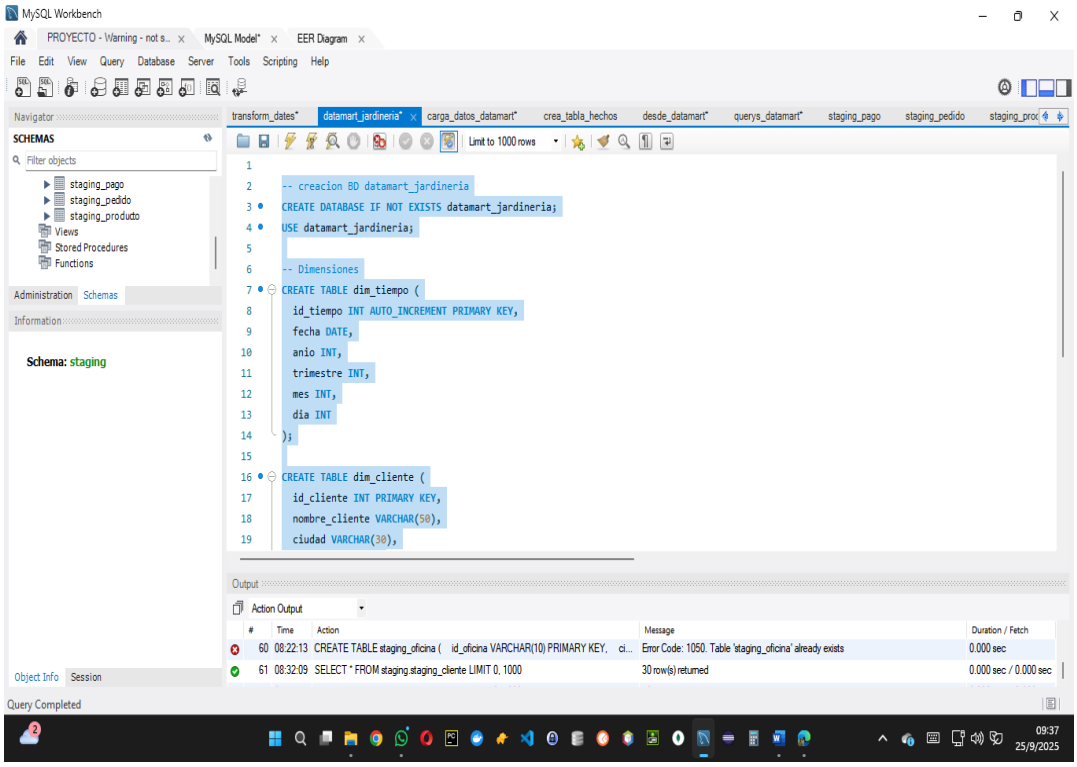


4. CARGA DE REGISTROS EN EL DATA MART FINAL:

CREACIÓN BD DATAMART

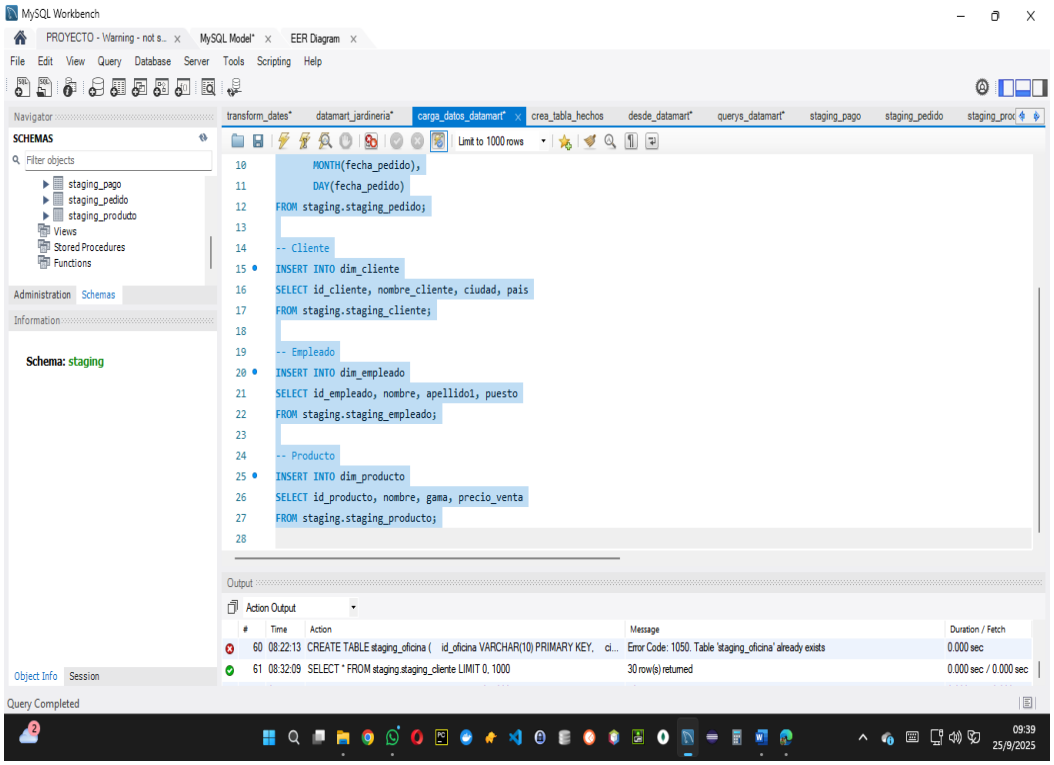
```
-- creacion BD datamart_jardineria
CREATE DATABASE IF NOT EXISTS datamart_jardineria;
USE datamart_jardineria;

-- Dimensiones
CREATE TABLE dim_tiempo (
  id_tiempo INT AUTO_INCREMENT PRIMARY KEY,
  fecha DATE,
  anio INT,
  trimestre INT,
  mes INT,
  dia INT
);

CREATE TABLE dim_cliente (
  id_cliente INT PRIMARY KEY,
  nombre_cliente VARCHAR(50),
  ciudad VARCHAR(30),
  pais VARCHAR(30)
);

CREATE TABLE dim_empleado (
```

```
  id_empleado INT PRIMARY KEY,
  nombre VARCHAR(30),
  apellido1 VARCHAR(30),
  puesto VARCHAR(50)
);

CREATE TABLE dim_producto (
  id_producto INT PRIMARY KEY,
  nombre VARCHAR(50),
  gama VARCHAR(50),
  precio_venta DECIMAL(15,2)
);
```



CARGA DE DATOS DE STAGING A BD DATAMART

```
-- carga de datos
USE datamart_jardineria;
-- Tiempo
INSERT INTO dim_tiempo (fecha, anio, trimestre, mes, dia)
SELECT DISTINCT fecha_pedido,
    YEAR(fecha_pedido),
    QUARTER(fecha_pedido),
    MONTH(fecha_pedido),
    DAY(fecha_pedido)
FROM staging.staging_pedido;

-- Cliente
INSERT INTO dim_cliente
SELECT id_cliente, nombre_cliente, ciudad, pais
FROM staging.staging_cliente;

-- Empleado
INSERT INTO dim_empleado
SELECT id_empleado, nombre, apellido1, puesto
FROM staging.staging_empleado;

-- Producto
INSERT INTO dim_producto
SELECT id_producto, nombre, gama, precio_venta
FROM staging.staging_producto;
```


MySQL Workbench
PROYECTO - Warning - not s...
MySQL Model*
EER Diagram
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
staging_pago
staging_pedido
staging_producto
Views
Stored Procedures
Functions
Administration Schemas
Information
Schema: staging
transform_dates*
datamart_jardineria*
carga_datos_datamart*
crea_tabla_hechos
desde_datamart*
querys_datamart*
staging_pago
staging_pedido
Limit to 1000 rows
MONTH(fecha_pedido),
DAY(fecha_pedido)
FROM staging.staging_pedido;
-- Cliente
INSERT INTO dim_cliente
SELECT id_cliente, nombre_cliente, ciudad, pais
FROM staging.staging_cliente;
-- Empleado
INSERT INTO dim_empleado
SELECT id_empleado, nombre, apellido1, puesto
FROM staging.staging_empleado;
-- Producto
INSERT INTO dim_producto
SELECT id_producto, nombre, gama, precio_venta
FROM staging.staging_producto;
Output
Action Output
Time Action Message Duration / Fetch
60 08:22:13 CREATE TABLE staging_oficina (id_oficina VARCHAR(10) PRIMARY KEY, ci... Error Code: 1050. Table 'staging_oficina' already exists 0.000 sec
61 08:32:09 SELECT * FROM staging.staging_cliente LIMIT 0, 1000 30 row(s) returned 0.000 sec / 0.000 sec
Object Info Session
Query Completed
09:39
25/9/2025

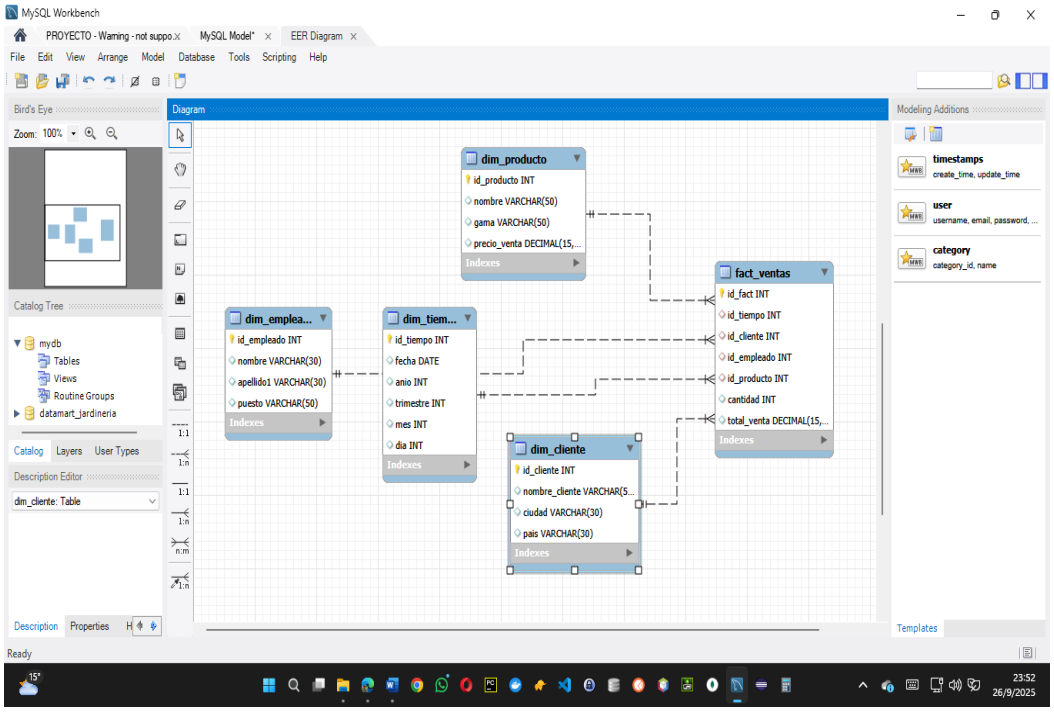

MySQL Workbench
PROYECTO - Warning - not suppo...
MySQL Model*
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Bird's Eye
Zoom: 100%
Catalog Tree
mydb
Tables
Views
Routine Groups
datamart_jardineria
Catalog Layers User Types
Description Editor
dim_cliente: Table
Description Properties
Diagram
dim_producto
id_producto INT
nombre VARCHAR(50)
gama VARCHAR(50)
precio_venta DECIMAL(15,...
Indexes
dim_emplea...
id_empleado INT
nombre VARCHAR(30)
apellido1 VARCHAR(30)
puesto VARCHAR(50)
Indexes
dim_tiem...
id_tiempo INT
fecha DATE
anio INT
trimestre INT
mes INT
dia INT
Indexes
fact_ventas
id_fact INT
id_tiempo INT
id_cliente INT
id_empleado INT
id_producto INT
cantidad INT
total_venta DECIMAL(15,...
Indexes
dim_cliente
id_cliente INT
nombre_cliente VARCHAR(5...
ciudad VARCHAR(30)
pais VARCHAR(30)
Indexes
Modeling Additions
timestamps
create_time, update_time
user
username, email, password, ...
category
category_id, name
Templates
Ready
23:52
26/9/2025

TABLA DE HECHOS

```
-- Tabla de hechos
CREATE TABLE fact_ventas (
  id_fact INT AUTO_INCREMENT PRIMARY KEY,
  id_tiempo INT,
  id_cliente INT,
  id_empleado INT,
  id_producto INT,
  cantidad INT,
  total_venta DECIMAL(15,2),
  FOREIGN KEY (id_tiempo) REFERENCES dim_tiempo(id_tiempo),
  FOREIGN KEY (id_cliente) REFERENCES dim_cliente(id_cliente),
  FOREIGN KEY (id_empleado) REFERENCES dim_empleado(id_empleado),
  FOREIGN KEY (id_producto) REFERENCES dim_producto(id_producto)
);
```

POBLAR TABLA DE HECHOS

```
USE datamart_jardineria;

INSERT INTO fact_ventas (
    id_tiempo,
    id_cliente,
    id_empleado,
    id_producto,
    cantidad,
    total_venta
)
```

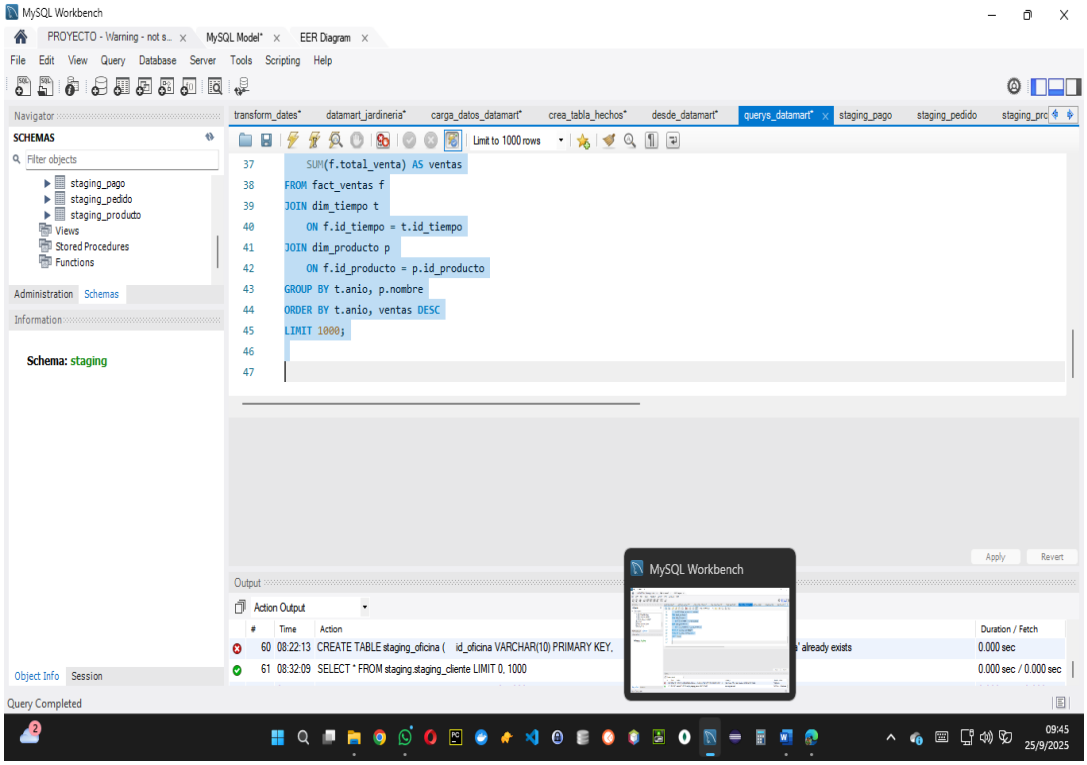
```
    SELECT
        t.id_tiempo,
        p.id_cliente,
        c.id_empleado_rep_ventas AS id_empleado,
        dp.id_producto,
        dp.cantidad,
        dp.cantidad * dp.precio_unidad AS total_venta
    FROM staging.staging_detalle_pedido dp
JOIN staging.staging_pedido p
        ON dp.id_pedido = p.id_pedido
    JOIN staging.staging_cliente c
        ON p.id_cliente = c.id_cliente
    JOIN dim_tiempo t
        ON p.fecha_pedido = t.fecha;
```

CONSULTAS PARA VERIFICACION DE DATOS CARGADOS A DATAMART

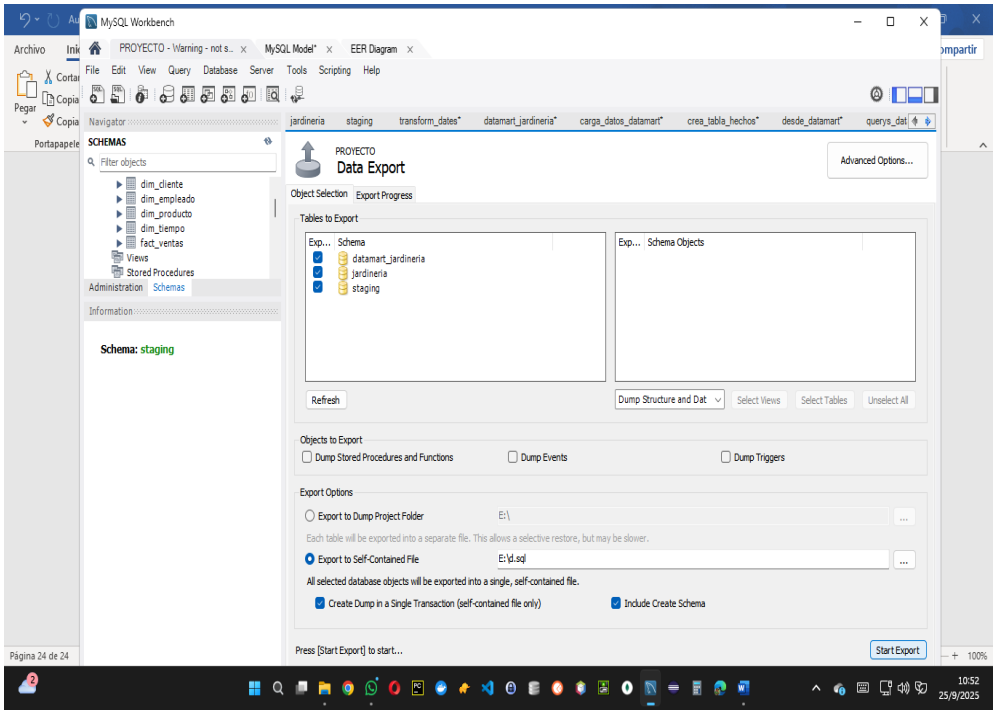
```
-- producto mas vendido
SELECT p.nombre AS producto,
   SUM(f.cantidad) AS total_vendido
FROM fact_ventas f
JOIN dim_producto p ON f.id_producto = p.id_producto
GROUP BY p.nombre
ORDER BY total_vendido DESC
LIMIT 1;

-- ventas totales por mes
```

```
SELECT t.anio, t.mes, SUM(f.total_venta) AS ventas_mes
FROM fact_ventas f
JOIN dim_tiempo t ON f.id_tiempo = t.id_tiempo
GROUP BY t.anio, t.mes
ORDER BY t.anio, t.mes;

-- mejor cliente por mas compras
SELECT c.nombre_cliente, SUM(f.total_venta) AS total_compras
FROM fact_ventas f
JOIN dim_cliente c ON f.id_cliente = c.id_cliente
GROUP BY c.nombre_cliente
ORDER BY total_compras DESC
LIMIT 1;

-- mejor vendedor
SELECT CONCAT(e.nombre, ' ', e.apellido1) AS empleado,
   SUM(f.total_venta) AS total_ventas
FROM fact_ventas f
JOIN dim_empleado e ON f.id_empleado = e.id_empleado
GROUP BY empleado
ORDER BY total_ventas DESC
LIMIT 1;

-- ventas por producto y año
SELECT t.anio,
   p.nombre AS producto,
```

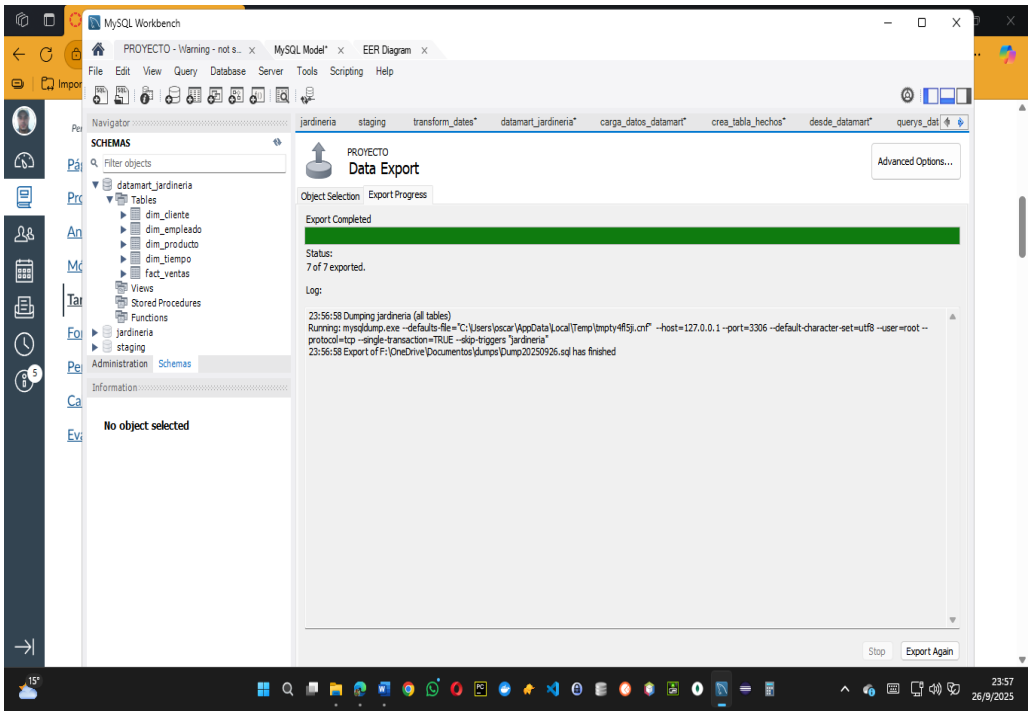
```
    SUM(f.total_venta) AS ventas
FROM fact_ventas f
JOIN dim_tiempo t
    ON f.id_tiempo = t.id_tiempo
JOIN dim_producto p
    ON f.id_producto = p.id_producto
GROUP BY t.anio, p.nombre
ORDER BY t.anio, ventas DESC
LIMIT 1000;
```



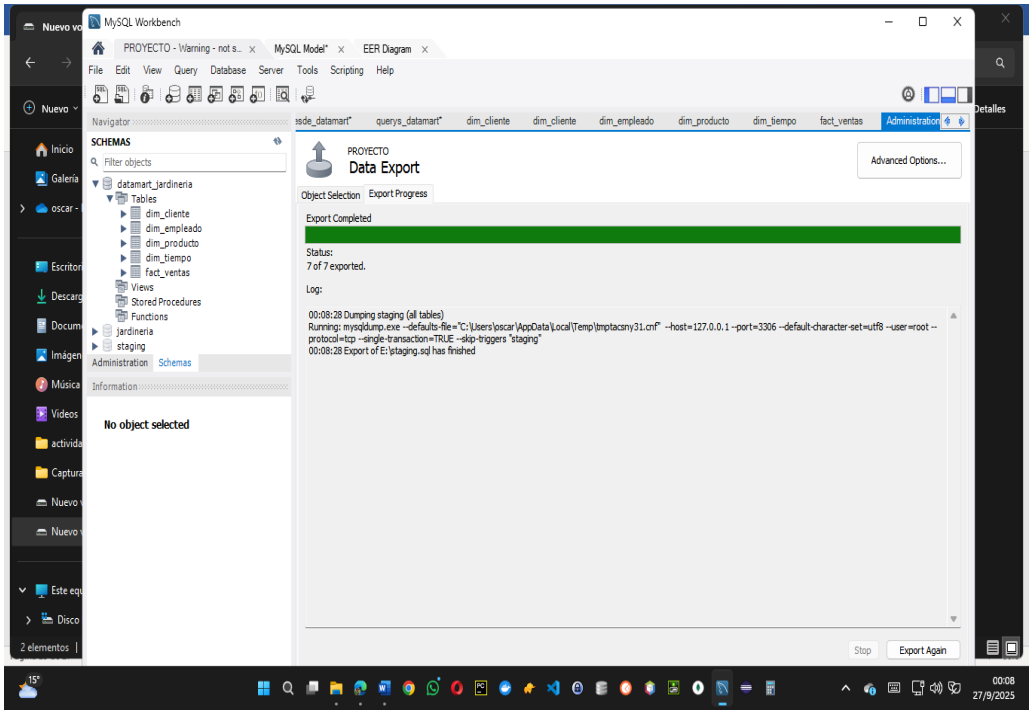
BACKUP BASES DE DATOS



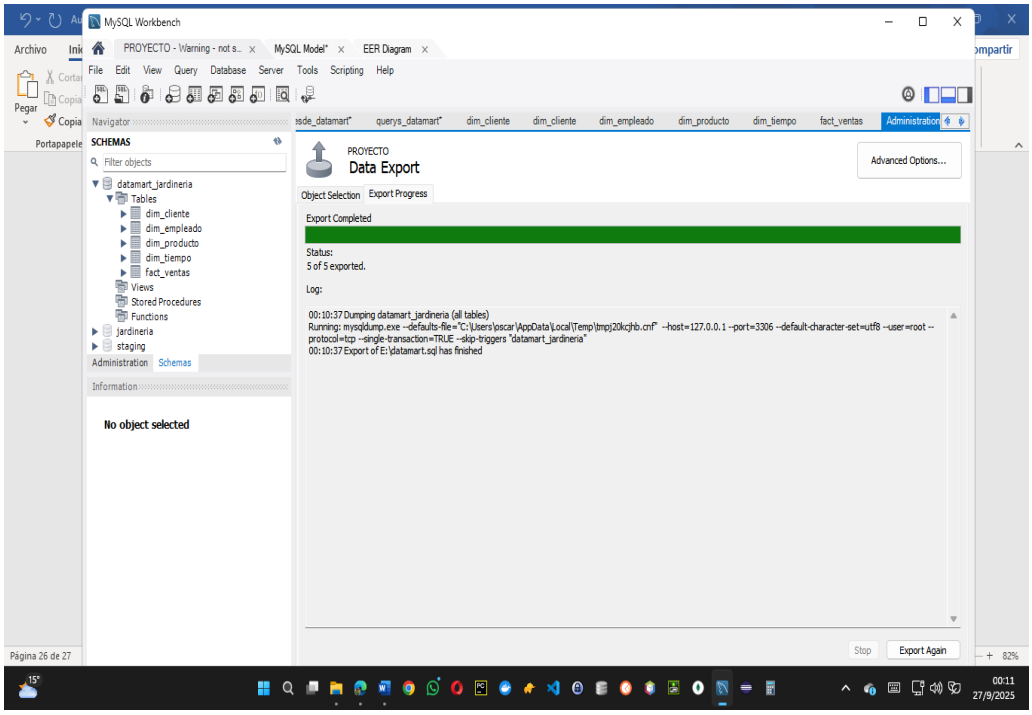
BD JARDINERIA



BD STAGING



BD DATAMART



CONCLUSIONES:

La implementación del proceso **ETL** permitió mejorar la calidad y consistencia de los datos, eliminando errores comunes como fechas incorrectas o valores nulos.

2. La construcción del **DATAMART** en un modelo en estrella facilita la consulta eficiente y el análisis multidimensional de la información.

3. El uso de **WORKBENCH** y **SQL** como herramienta principal demostró ser adecuado para realizar la limpieza, transformación y carga de datos de forma controlada y reproducible.

BIBLIOGRAFIA

Vista de LA INTEGRACIÓN DE DATAMART CON DATAWAREHOUSE. (2025). Retrieved 27 September 2025, from https://revistas.unesum.edu.ec/index.php/unesumciencias/article/view/470/501