- **Proyecto Final:** Curso SQL.

- **Comisión:** 39790.

- **Profesor:** Maximiliano Ibarra.

- **Tutor :** Ariel Annone.

- **Alumno:** Pablo Alessandrini.

# Introducción:

El presente proyecto tiene como objetivo el diseño e implementación de una base de datos para una empresa minorista del sector de electrónica y electrodomésticos. El enfoque principal se centra en gestionar de manera eficiente los datos relacionados con las compras de los clientes, los montos de las ventas y las interacciones entre los vendedores y los clientes. Esta base de datos permitirá mejorar la toma de decisiones estratégicas, optimizar la gestión del personal y ofrecer una experiencia personalizada a los clientes.

# Situación problemática:

La empresa se enfrenta a diversos desafíos en su gestión diaria, como la falta de información detallada sobre las compras de los clientes, la dificultad para identificar a los vendedores más efectivos y la necesidad de tener un seguimiento más preciso de los montos de las ventas. Estas problemáticas limitan la capacidad de la empresa para identificar oportunidades de crecimiento, asignar recursos de manera efectiva y ofrecer un servicio personalizado a sus clientes.

# Objetivo:

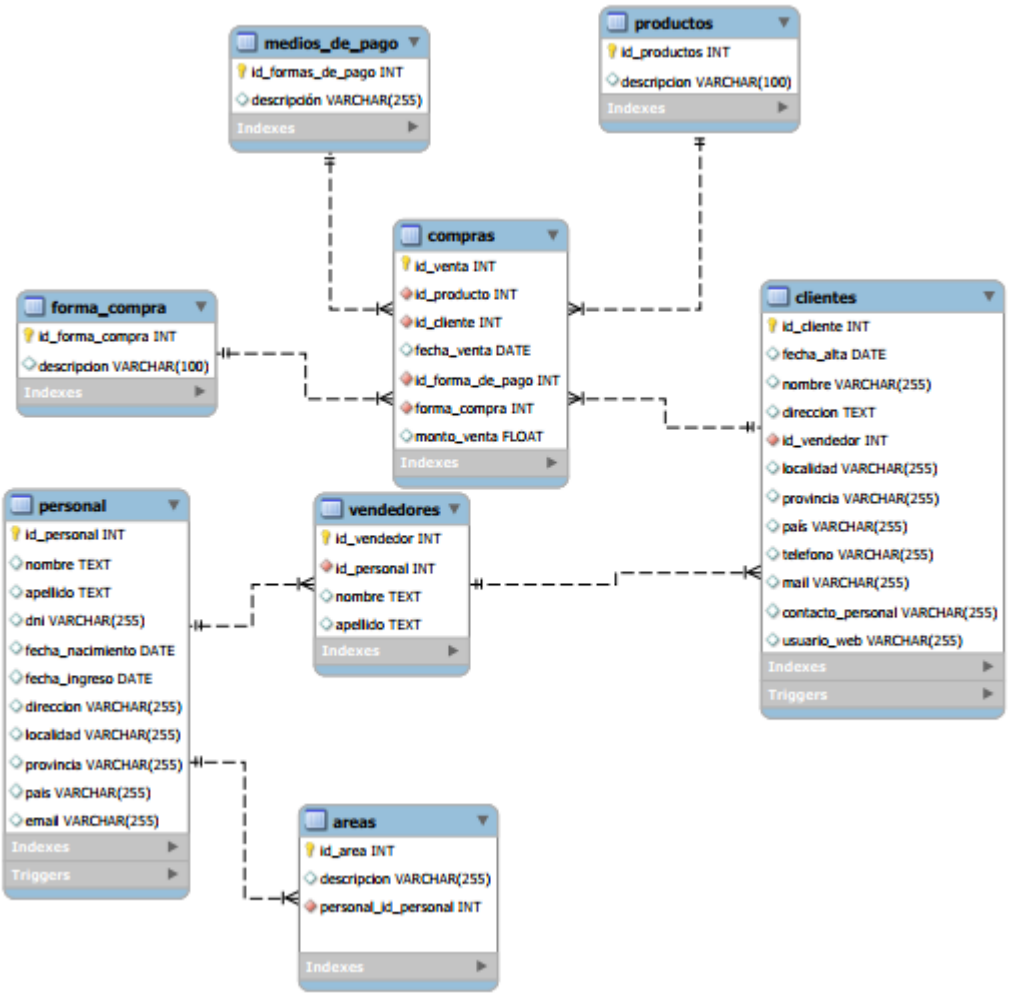
El objetivo principal de esta base de datos es brindar a la empresa una herramienta integral para gestionar y analizar los datos relacionados con las compras de los clientes, los montos de las ventas y las interacciones entre los vendedores y los clientes. Se busca mejorar la toma de decisiones estratégicas al identificar patrones de compra, evaluar el rendimiento de los vendedores y segmentar a los clientes en base a sus preferencias y comportamientos de compra.

# Modelo de negocio:

La empresa se destaca por ofrecer una amplia variedad de productos de electrónica y electrodomésticos de alta calidad, así como un excelente servicio al cliente. Su modelo de negocio se basa en brindar una experiencia personalizada a cada cliente, ofreciendo asesoramiento especializado, promociones exclusivas y una atención personalizada. La empresa cuenta con un equipo de vendedores capacitados que se esfuerzan por establecer relaciones sólidas con los clientes y aumentar las ventas.

# Diagrama E-R:

El diagrama Entidad-Relación (E-R) representa las entidades principales y las relaciones entre ellas en la base de datos. En este caso, el diagrama E-R incluirá entidades como Clientes, Compras, Vendedores y Ventas, junto con sus atributos y las relaciones que existen entre ellas. Este diagrama servirá como guía para la creación de las tablas y consultas SQL necesarias para gestionar eficientemente los datos relacionados con las compras, los montos de las ventas y las interacciones entre vendedores y clientes.



## DESCRIPCIÓN DE LAS TABLAS DE LA  BASE DE DATOS.

- **TABLA ÁREA**

La siguiente tabla proporciona una descripción de las áreas existentes en la empresa, junto con sus respectivos nombres.

| TABLA ÁREA | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_area | identificador del área | int | PK |
| descripción | detalla cómo se llama el área | varchar(255) | |

- **TABLA CLIENTES**

La tabla "Clientes" contiene información clave sobre los clientes de la empresa, como su identificación, nombre, dirección y detalles de contacto. También se registra el vendedor asignado para un mejor seguimiento.

| TABLA CLIENTES | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_cliente | identificador del clientes (estilo legajo) | int | **PK** |
| fecha_alta | fecha de que se ingresó a la base | date | |
| nombre | Nombre del cliente | varchar(255) | |
| dirección | dirección del cliente | text | |
| id_vendedor | identificador del vendedor que tiene cada cliente | int | **FK** (referenciada a Tabla Vendedores) |
| localidad | localidad del cliente | varchar(255) | |
| provincia | provincia del cliente | varchar(255) | |

| **TABLA CLIENTES** | | | |
|---|---|---|---|
| país | país del cliente | varchar(255) | |
| teléfono | teléfono del cliente | varchar(255) | |
| mail | email del cliente | varchar(255) | |
| contacto | persona del cliente que sirve de contacto | varchar(255) | |
| usuario_web | usuario del cliente para acceso a la web | varchar(255) | |

- **TABLA COMPRAS**

La tabla "Compras" registra información sobre las ventas de la empresa, incluyendo el identificador de venta, el producto vendido, el cliente, la fecha, la forma de pago, la forma de compra y el monto.

| **TABLA COMPRAS** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_venta | identificador de cada venta | int | PK |
| id_producto | identificador del producto de la venta | int | **FK** (referenciada a Tabla Productos) |
| id_cliente | identificador del cliente que efectuó la compra | int | **FK** (referenciada a Tabla Clientes) |
| fecha_venta | fecha de la compra | date | |
| id_forma_de_pago | identificador de la forma que se pagó | int | **FK** (referenciada a Tabla Formas de pago) |
| forma_compra | como se hizo la compra (presencial o web) | int | **FK** (referenciada a Tabla Forma de compra) |
| monto_venta | precio/monto de la compra hecha | float | |

- **TABLA FORMA DE COMPRA**

La tabla "Forma de Compra" almacena información sobre las diferentes formas de compra utilizadas por los clientes(web o presencial), incluyendo el identificador de la forma de compra y su descripción.

| TABLA FORMA COMPRAS | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_forma_compra | identificador de la forma que se efectuó la compra (presencial/web) | int | PK |
| descripción | descripción de la forma de la compra (presencial/web) | varchar(100) | |

- **TABLA MEDIOS DE PAGO.**

La tabla "Medios de Pago" registra los diferentes métodos utilizados para pagar las compras.

| TABLA MEDIOS DE PAGO | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_formas_de_pago | identificador de cómo se efectuó el pago | int | PK |
| descripción | tipo de forma de pago (ej. efectivo, tarjeta) | varchar(255) | |

- **TABLA PERSONAL.**

La tabla "Personal" contiene información básica sobre el personal de la empresa, como nombres, apellidos, DNI, fechas de nacimiento e ingreso, dirección, localidad, provincia, país y correo electrónico. Es utilizada para gestionar y mantener los datos del personal de la empresa de manera organizada.

| **TABLA PERSONAL** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_personal | identificador de cada empleado ingresado a la base | int | **PK** |
| id_area | identificador del área perteneciente | int | **FK** (referenciada a Tabla Áreas) |
| nombre | nombre del empleado | text | |
| apellido | apellido del empleado | text | |
| dni | documento del empleado | varchar(255) | |
| fecha_nacimiento | fecha de nacimiento del empleado | date | |
| fecha_ingreso | fecha de ingreso a la empresa | date | |
| dirección | dirección donde tiene locación el empleado | varchar(255) | |
| localidad | localidad donde es el empleado | varchar(255) | |
| provincia | provincia donde es el empleado | varchar(255) | |
| país | país donde es el empleado | varchar(255) | |
| email | email del empleado | varchar(255) | |

- **TABLA PRODUCTOS.**

La tabla "Productos" almacena información sobre los productos de la empresa, incluyendo un identificador único para cada producto, una descripción del producto y el tipo al que pertenece. Esta tabla permite gestionar y mantener los datos de los productos de manera estructurada.

| **TABLA PRODUCTOS** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_productos | identificador de cada producto | int | PK |
| descripción | descripción de qué tipo de producto es | varchar(100) | |

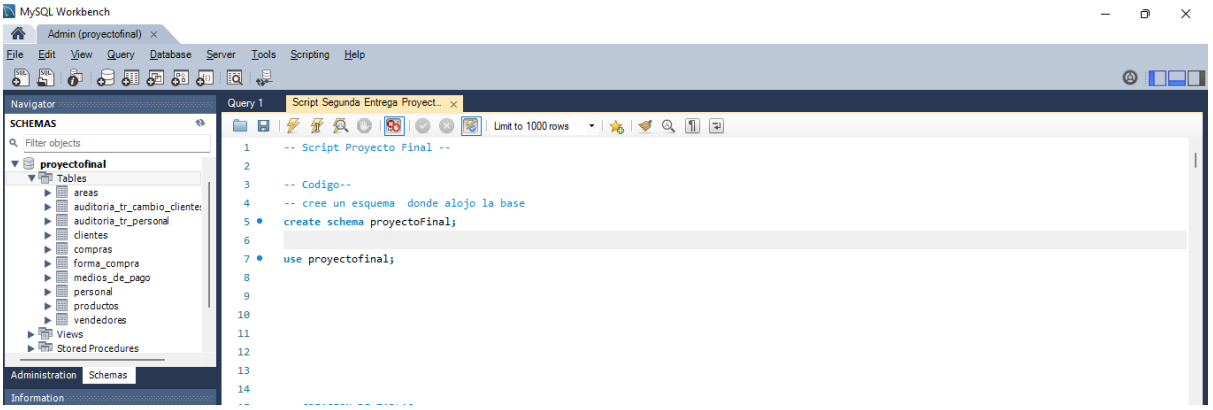
- **TABLA VENDEDORES.**

La tabla "Vendedores" registra información de los vendedores de la empresa, incluyendo un identificador único para cada vendedor, el identificador del personal asociado, el nombre y el apellido del vendedor.

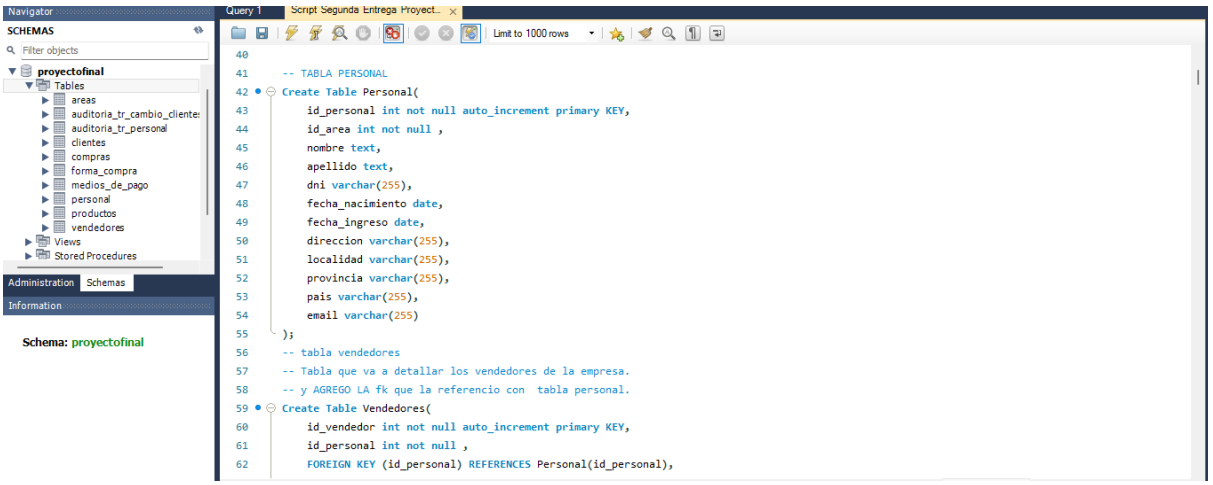
| **TABLA VENDEDORES** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_vendedor | identificador de cada vendedor | int | PK |
| id_personal | identificador del área vendedores | int | **FK** (referenciada a Tabla Personal) |
| nombre | nombre del vendedor | text | |
| apellido | apellido del vendedor | text | |

# Descripción de Scripts utilizados

- **Creación de la base de datos y esquema:**
    - Script para crear la estructura básica de la base de datos que usaremos en el Proyecto Final.
    - Definición del esquema de la base de datos, incluyendo tablas, relaciones y restricciones.

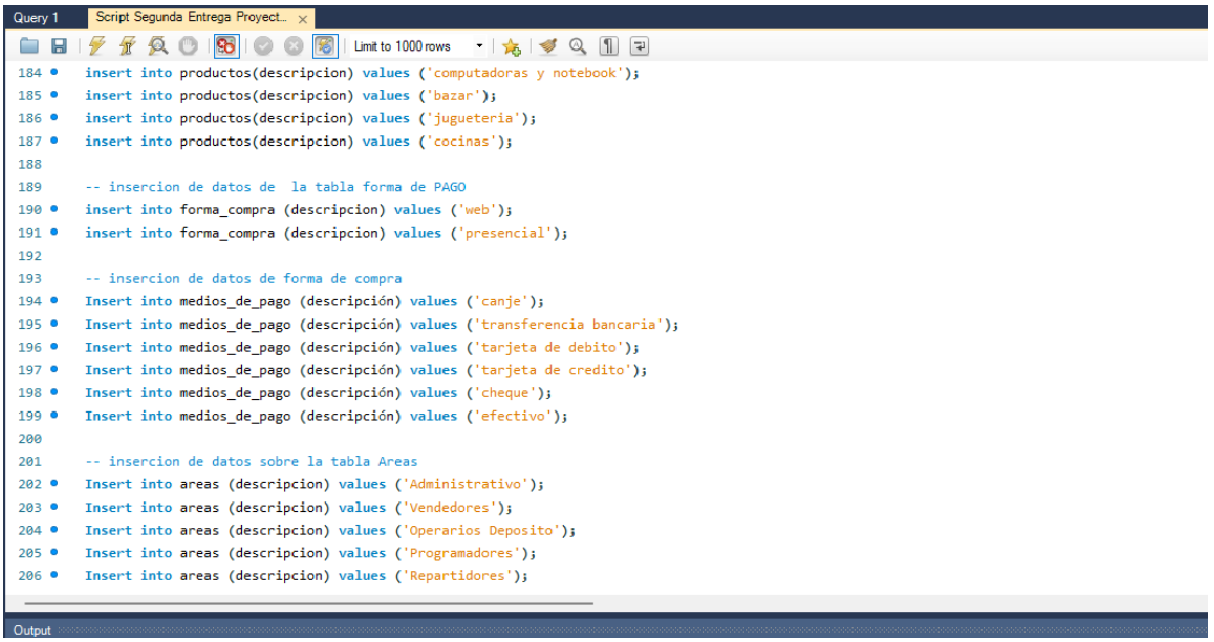



- **Creación de tabla:**
    - Creación de tabla: Script para crear tablas individuales en la base de datos. Por ejemplo, en el caso de la Tabla Personal y Vendedores, se incluirán los nombres de las columnas, el tipo de dato que albergarán y las llaves correspondientes (ya sean foráneas o primarias).

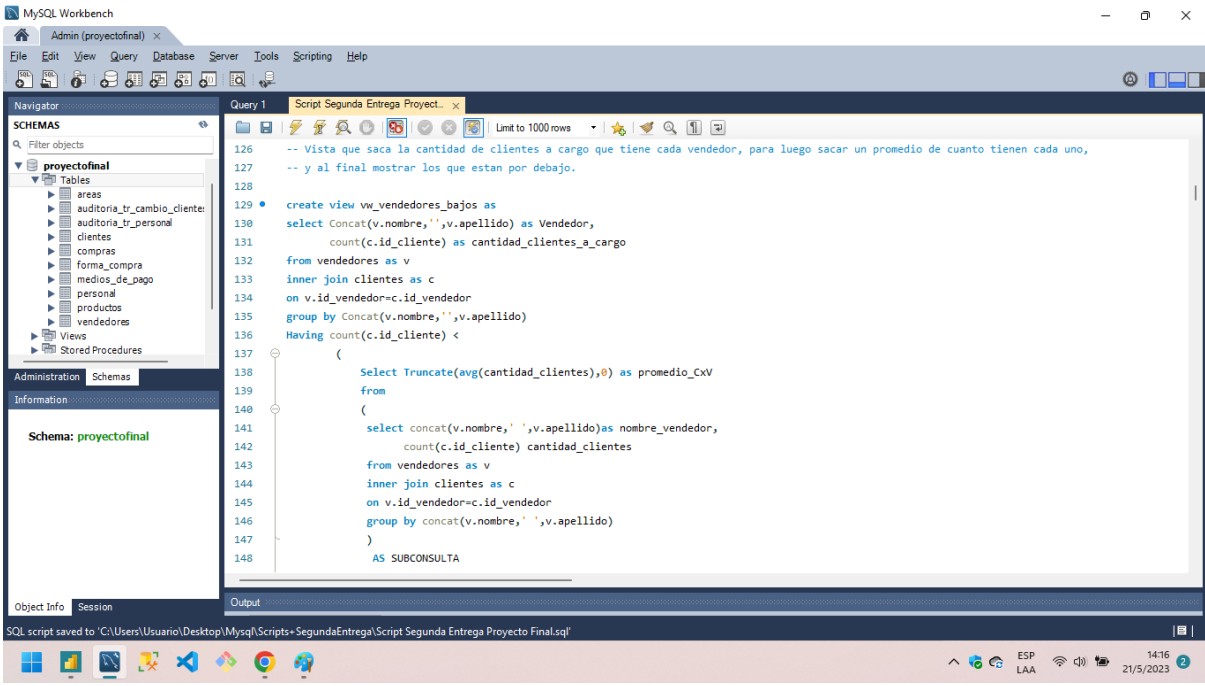



- **<u>Inserción de datos:</u>**

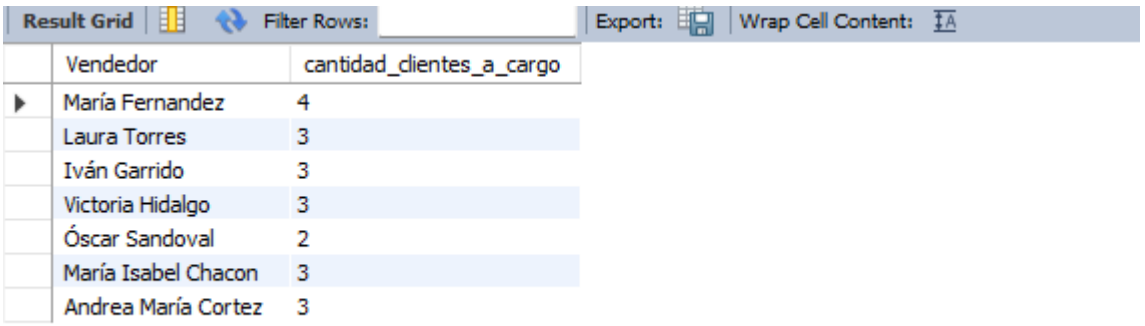
  - Inserción de datos iniciales: Scripts para insertar datos iniciales en las tablas. A continuación, se muestra cómo se poblaron algunas de las tablas utilizando el método manual de inserción de datos. Por ejemplo, se incluyen ejemplos de la Tabla Productos, Formas de Pago y Formas de Compra.



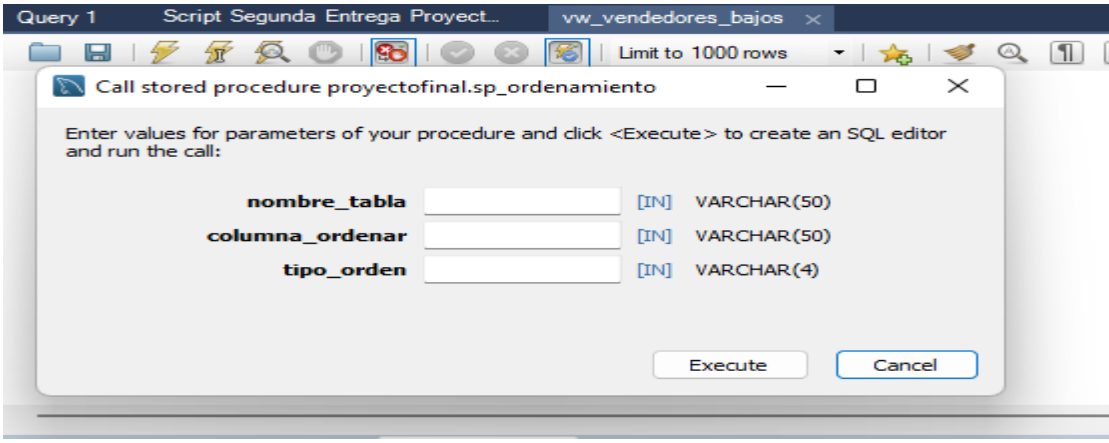
- **Creación de vistas:**
  - Se crearon para proporcionar diferentes perspectivas de los datos.
    En pre entregas anteriores, se desarrollaron vistas con enfoque gerencial para el control de los empleados, específicamente para los vendedores
    Por ejemplo, se incluye una vista que evalúa qué vendedores tienen un bajo promedio de cantidad de clientes a cargo. A continuación, se detalla la estructura y los criterios utilizados en la misma.
    De esta manera, se podrá destacar en un informe la importancia de las vistas en el análisis y seguimiento de los vendedores, así como proporcionar un ejemplo concreto de una vista que evalúa el desempeño de los mismos en función de la cantidad de clientes que atienden.



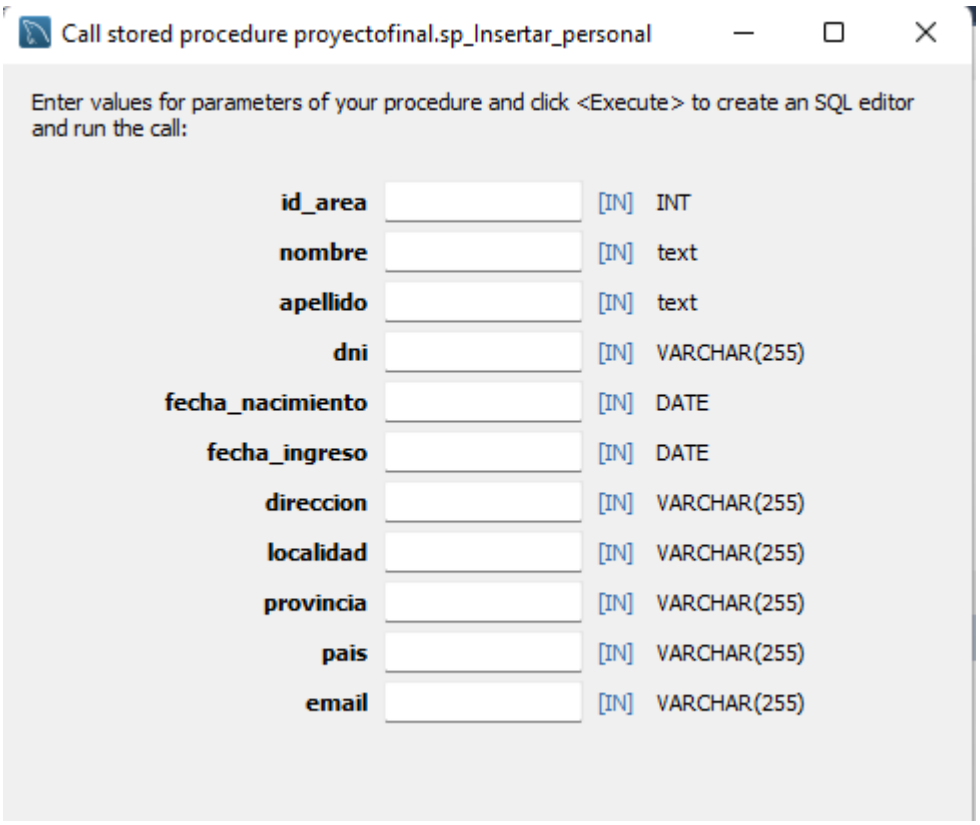

Y acá podemos obtener el resultado de la consulta anterior, para poder tomar decisiones consecuentes.



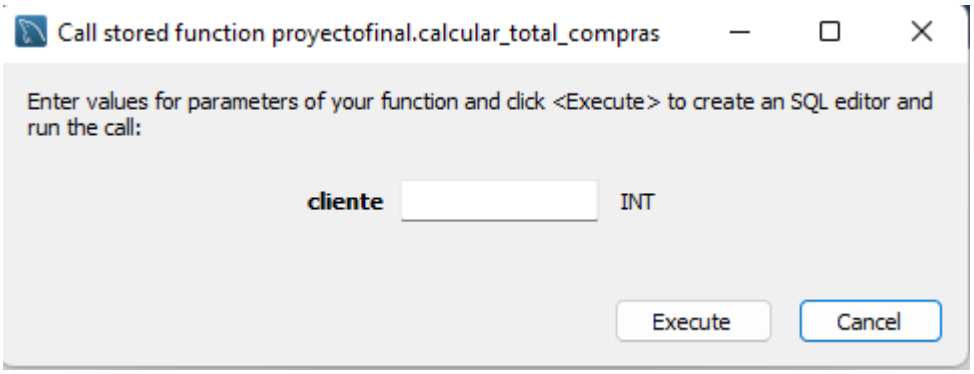

| Vendedor | cantidad_clientes_a_cargo |
|---|---|
| María Fernandez | 4 |
| Laura Torres | 3 |
| Iván Garrido | 3 |
| Victoria Hidalgo | 3 |
| Óscar Sandoval | 2 |
| María Isabel Chacon | 3 |
| Andrea María Cortez | 3 |

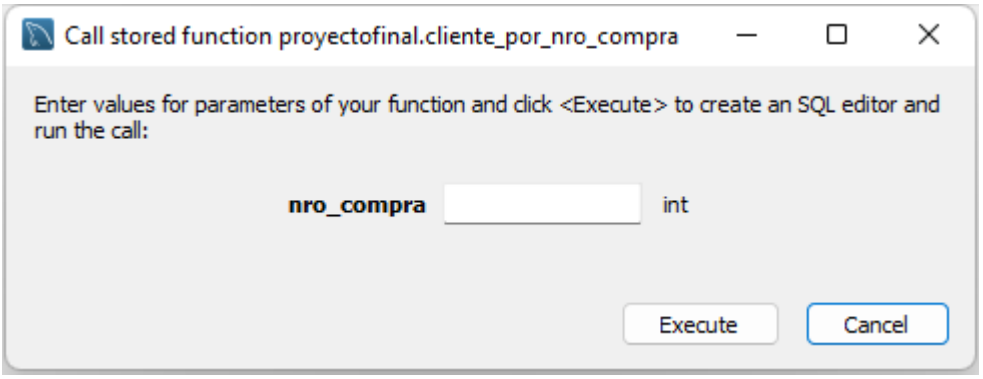
- **<u>Creación de stored procedures:</u>**
    - Se han realizado 2
        - Ordenamiento de columna. Acá lo que tenemos es un procedimiento almacenado que ingresando el nombre de la tabla, podemos ordenar según una columna y en orden ascendente o descendente.
        - 


        - **<u>Inserción de datos.</u>**
        - Se desarrolló un procedimiento para automatizar y optimizar la inserción de datos en la tabla "Personal". Este procedimiento permite una carga eficiente y precisa de los datos, simplificando el proceso de inserción y garantizando la integridad de la información.

- 


- **<u>Creación de funciones:</u>**
  - Se han creado dos funciones que realizan cálculos y procedimientos específicos. En el primer caso, al ingresar el número de cliente, la función devuelve la cantidad de dinero que ha generado en compras dentro de la empresa. En el segundo caso, al proporcionar el número de compra, la función identifica qué cliente realizó dicha transacción.






- **Creación de triggers:**
  - Scripts que se ejecutan automáticamente cuando ocurren eventos específicos en la base de datos.
    Se han creado dos triggers para llevar un registro de eventos específicos en la base de datos. El primer trigger se encarga de registrar la información del personal ingresado, utilizando un stored procedure previamente creado. El segundo trigger se ha implementado para mantener un registro de los cambios en los vendedores a cargo, respecto a los clientes. Estos triggers garantizan la integridad de los datos y proporcionan un historial de eventos relevantes para su posterior consulta y análisis.

    - **tr_registros_ingresos_personal_AI;**
    - **tr_registros_cambios_cliente_vendedor_BU;**

  - **Tablas de auditoría:**
    - Creación de tablas adicionales para el seguimiento de cambios y auditoría.
    - Para poder llevar un registro de los triggers creados, se generaron las dos tablas auditorías.
    - Tabla que controla ingresos de personal a la base de datos
      - →auditoria_tr_personal

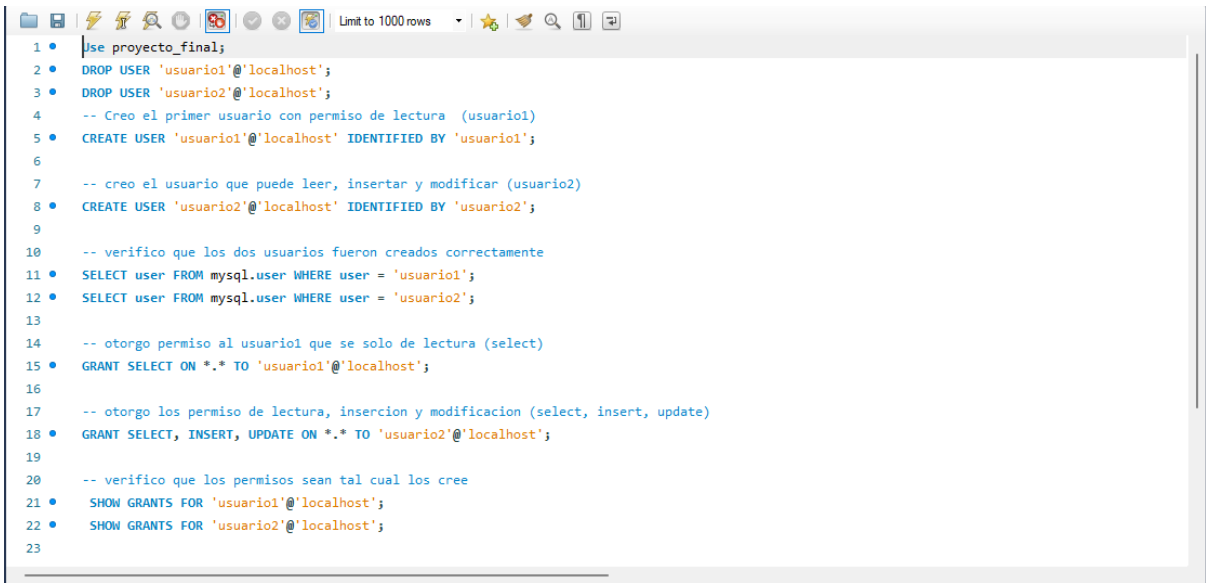
| **TABLA “auditoria_tr_cambio_clientes_a_cargo”** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_tr | identificador del número de Triggers | int | PK |
| id_area | id del área del nuevo ingresante | int | |
| nombre | nombre del nuevo ingresante | text | |
| apellido | apellido del nuevo ingresante | text | |
| dni | dni del nuevo ingresante | varchar(255) | |
| email | email del nuevo ingresante | varchar(255) | |
| fecha_accion | fecha de cuando se ejecutó la acción ( Ej Insert/ingreso a la base de datos) | datetime | |
| usuario | que usuario ingreso la acción | varchar(255) | |
| acción | tipo de acción ejecutada | varchar(255) | |

- Tabla que controla cambios de cliente
  →auditoria_tr_cambio_clientes_a_cargo

| **TABLA “auditoria_tr_cambio_clientes_a_cargo”** | | | |
|---|---|---|---|
| **COLUMNA** | **DESCRIPCIÓN** | **TIPO DE DATOS** | **TIPO DE CLAVE** |
| id_tr | identificador del número de Triggers | int | PK |
| nombre | nombre del cliente | varchar(255) | |
| id_vendedor | identificador del cliente | int | |
| mail | email del cliente | varchar(255) | |
| nuevo_id_vendedor | id del nuevo vendedor que tendrá a cargo | int | |
| usuario | qué usuario hizo el cambio | varchar(255) | |
| fecha_accion | cuando se hizo el cambio | datetime | |
| accion | tipo de acción que se ejecutó( aca update) | varchar(255) | |

- **Creación de usuarios:**
- Se han desarrollado scripts para la creación de usuarios y la asignación de permisos de acceso en la base de datos. Se han creado dos usuarios con distintos niveles de acceso:
  - Usuario 1: Se le ha otorgado permiso de lectura (SELECT) en la base de datos.
  - Usuario 2: Se le ha otorgado permisos de lectura, inserción y actualización (SELECT/INSERT/UPDATE) en la base de datos.

  En ambos casos, se ha restringido el permiso de eliminación de registros. Estas configuraciones de acceso garantizan la seguridad y el control adecuado de los datos en la base de datos.



```
Use proyecto_final;
DROP USER 'usuario1'@'localhost';
DROP USER 'usuario2'@'localhost';
-- Creo el primer usuario con permiso de lectura  (usuario1)
CREATE USER 'usuario1'@'localhost' IDENTIFIED BY 'usuario1';

-- creo el usuario que puede leer, insertar y modificar (usuario2)
CREATE USER 'usuario2'@'localhost' IDENTIFIED BY 'usuario2';

-- verifico que los dos usuarios fueron creados correctamente
SELECT user FROM mysql.user WHERE user = 'usuario1';
SELECT user FROM mysql.user WHERE user = 'usuario2';

-- otorgo permiso al usuario1 que se solo de lectura (select)
GRANT SELECT ON *.* TO 'usuario1'@'localhost';

-- otorgo los permiso de lectura, insercion y modificacion (select, insert, update)
GRANT SELECT, INSERT, UPDATE ON *.* TO 'usuario2'@'localhost';

-- verifico que los permisos sean tal cual los cree
 SHOW GRANTS FOR 'usuario1'@'localhost';
 SHOW GRANTS FOR 'usuario2'@'localhost';
```

En los archivos adjuntos, se envía un Script, comprobando que los acceso otorgados funciones correctamente.

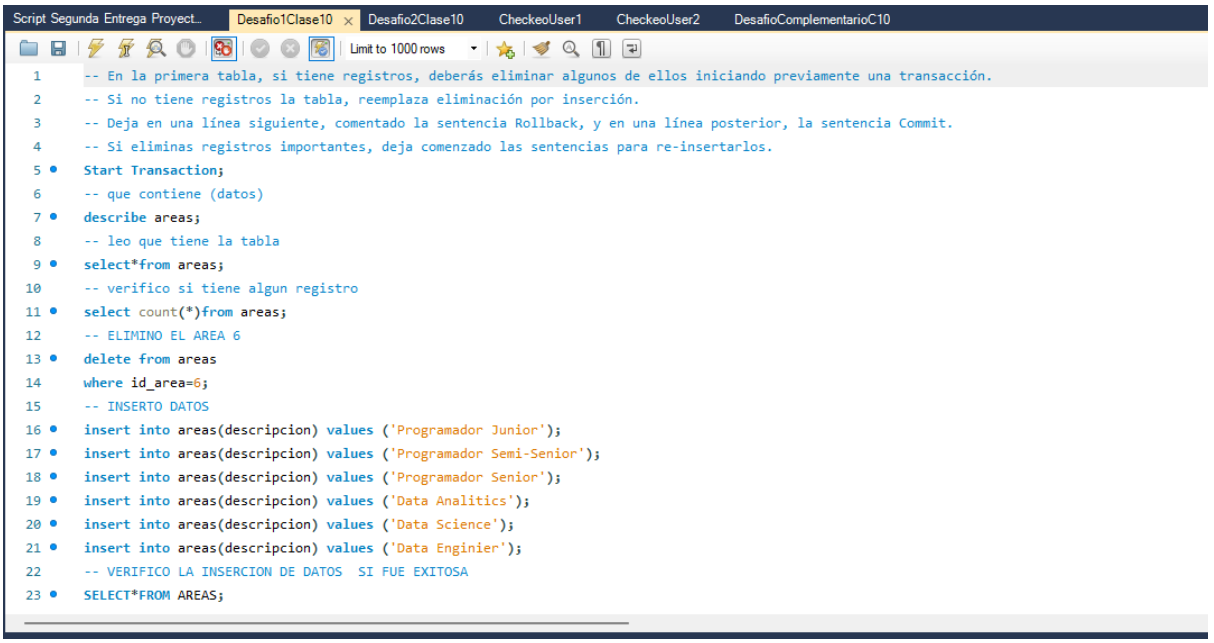
- **<u>Transacciones:</u>**
  - Uso de sentencias de transacción para garantizar la integridad y consistencia de los datos.
    Se utilizaron transacciones en dos escenarios:

  - 1. Eliminación de registros en la primera tabla:
    - Antes de realizar la eliminación, se inició una transacción para asegurar que los cambios fueran reversibles en caso de ser necesario. Se proporcionó la opción de realizar un rollback para deshacer los cambios o un commit para confirmarlos.

    2. Inserción de registros en la segunda tabla:
    - Se realizó la inserción de ocho nuevos registros dentro de una transacción. Además, se agregaron savepoints después de la inserción del cuarto y octavo registro para marcar puntos de guardado dentro de la transacción. También se incluyó la opción de eliminar los savepoints de los primeros cuatro registros insertados, aunque se dejó comentada para evitar su ejecución accidental.

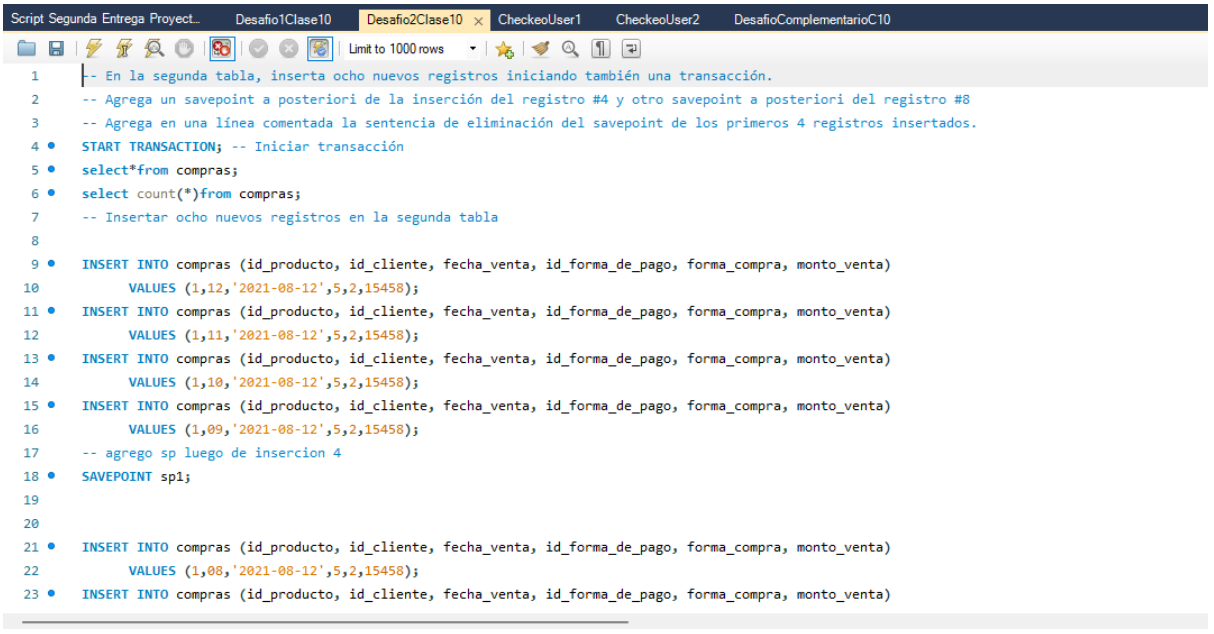
  - En ambos casos, el uso de transacciones permite controlar y garantizar la consistencia de los datos, proporcionando la capacidad de deshacer o confirmar los cambios realizados.

## Transacción Número 1 (Star Transaccion, Rollback y Commits)




```sql
-- En la primera tabla, si tiene registros, deberás eliminar algunos de ellos iniciando previamente una transacción.
-- Si no tiene registros la tabla, reemplaza eliminación por inserción.
-- Deja en una línea siguiente, comentado la sentencia Rollback, y en una línea posterior, la sentencia Commit.
-- Si eliminas registros importantes, deja comenzado las sentencias para re-insertarlos.
Start Transaction;
-- que contiene (datos)
describe areas;
-- leo que tiene la tabla
select*from areas;
-- verifico si tiene algun registro
select count(*)from areas;
-- ELIMINO EL AREA 6
delete from areas
where id_area=6;
-- INSERTO DATOS
insert into areas(descripcion) values ('Programador Junior');
insert into areas(descripcion) values ('Programador Semi-Senior');
insert into areas(descripcion) values ('Programador Senior');
insert into areas(descripcion) values ('Data Analitics');
insert into areas(descripcion) values ('Data Science');
insert into areas(descripcion) values ('Data Enginier');
-- VERIFICO LA INSERCION DE DATOS  SI FUE EXITOSA
SELECT*FROM AREAS;
```

## Transacción Número 2 (Star Transacción, Savepoints,  Rollback y Commits)




```sql
-- En la segunda tabla, inserta ocho nuevos registros iniciando también una transacción.
-- Agrega un savepoint a posteriori de la inserción del registro #4 y otro savepoint a posteriori del registro #8
-- Agrega en una línea comentada la sentencia de eliminación del savepoint de los primeros 4 registros insertados.
START TRANSACTION; -- Iniciar transacción
select*from compras;
select count(*)from compras;
-- Insertar ocho nuevos registros en la segunda tabla

INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,12,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,11,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,10,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,09,'2021-08-12',5,2,15458);
-- agrego sp luego de insercion 4
SAVEPOINT sp1;


INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,08,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
```

```sql

INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,08,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,07,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,06,'2021-08-12',5,2,15458);
INSERT INTO compras (id_producto, id_cliente, fecha_venta, id_forma_de_pago, forma_compra, monto_venta)
      VALUES (1,05,'2021-08-12',5,2,15458);
-- Agregar un savepoint después de la inserción del registro #8
SAVEPOINT sp2;

select count(*)from compras;

-- Agregar en una línea comentada la sentencia de eliminación del savepoint de los primeros 4 registros insertados.
ROLLBACK TO sp1;

-- Confirmar los cambios
COMMIT;
```

- **<u>Backup:</u>**
  - Script para realizar copias de seguridad de la base de datos.
    Se realizó la creación de un backup del proyecto final, siguiendo los siguientes aspectos clave:

    - Contenido del backup: El backup incluye únicamente las tablas de la base de datos, excluyendo las vistas. Se enfoca en respaldar los datos almacenados en las tablas, dejando de lado la estructura de la base de datos.

    - Dump Data Only: El enfoque principal del backup es guardar únicamente los datos de las tablas, sin incluir la estructura de las mismas. Esto implica que el backup se centra en preservar la información almacenada en las tablas, permitiendo su posterior restauración.

El objetivo de este proceso de creación de backup es asegurar la disponibilidad y protección de los datos del proyecto final. Al contar con un respaldo de los datos en un formato adecuado, se garantiza la posibilidad de recuperar la información en caso de pérdida o corrupción de la base de datos principal.

Script Segunda Entrega Proyect... | Desafio1Clase10 | Desafio2Clase10 | CheckeoUser1 | CheckeoUser2 | DesafioComplementarioC10 | Backup07052023

Limit to 1000 rows

```
1  -- Creacion de Backup Proyecto Final
2  -- Se incluyen solo las tablas (no las views)
3  -- Dump Data Only (solo los datos, dejando de lado la estructura, como asi lo pide la consigna)
4  /*Nombre de Tablas para validar--
5  areas, auditoria_tr_cambio_clientes, auditoria_tr_personal, clientes,compras, forma_compras,medios_de_pago,personal,productos,vendedores-- total
6  */
7
8  CREATE DATABASE  IF NOT EXISTS `proyectofinal` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION=
9  USE `proyectofinal`;
10 -- MySQL dump 10.13  Distrib 8.0.33, for Win64 (x86_64)
11 --
12 -- Host: localhost    Database: proyectofinal
13 -- ------------------------------------------------------
14 -- Server version   8.0.33
15
16 /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
17 /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
18 /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
19 /*!50503 SET NAMES utf8 */;
20 /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
21 /*!40103 SET TIME_ZONE='+00:00' */;
22 /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
23 /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
```

***La herramienta utilizada fue Mysql Workbench versión 8.0***