Universidad Autónoma de Baja California
Facultad de Contaduría y Administración

Inteligencia de Negocios

Base de Datos

Proyecto Final

-Integrantes:

- **De La Cruz Ramirez Jeremy Yael**
- **Esquivel Zarate Enrique**
- **Ramirez Cardenas Luis Armando**

-Grupo: 951

-Fecha: 30/05/2025

-Profesor: Fernando Christian Gandarilla Carrillo

Empresa seleccionada: Psicodelia Smoke Shop

Quienes somos:

Fundamos Psycodelia en el año 2014, con una mision: ser una marca, un estilo de vida. Creemos fielmente en nuestro movimiento. Nuestra pasion por la excelencia nos condujo a materializar esta mision, siendo la parte fundamental que nos ha impulsado a seguir adelante

En Psycodelia Tijuana, nos enorgullecemos de ofrecer a nivel nacional todos los productos relacionados con el smoke shop, vape shop y cbd a mayoreo y menudeo. Nuestra relacion de largo plazo, que hemos construido con nuestros clientes, nos respaldan lo cual nos ha permitido posicionarnos como el mejor smoke shop Tijuana

Diagrama Entidad - Relación(Estructura):

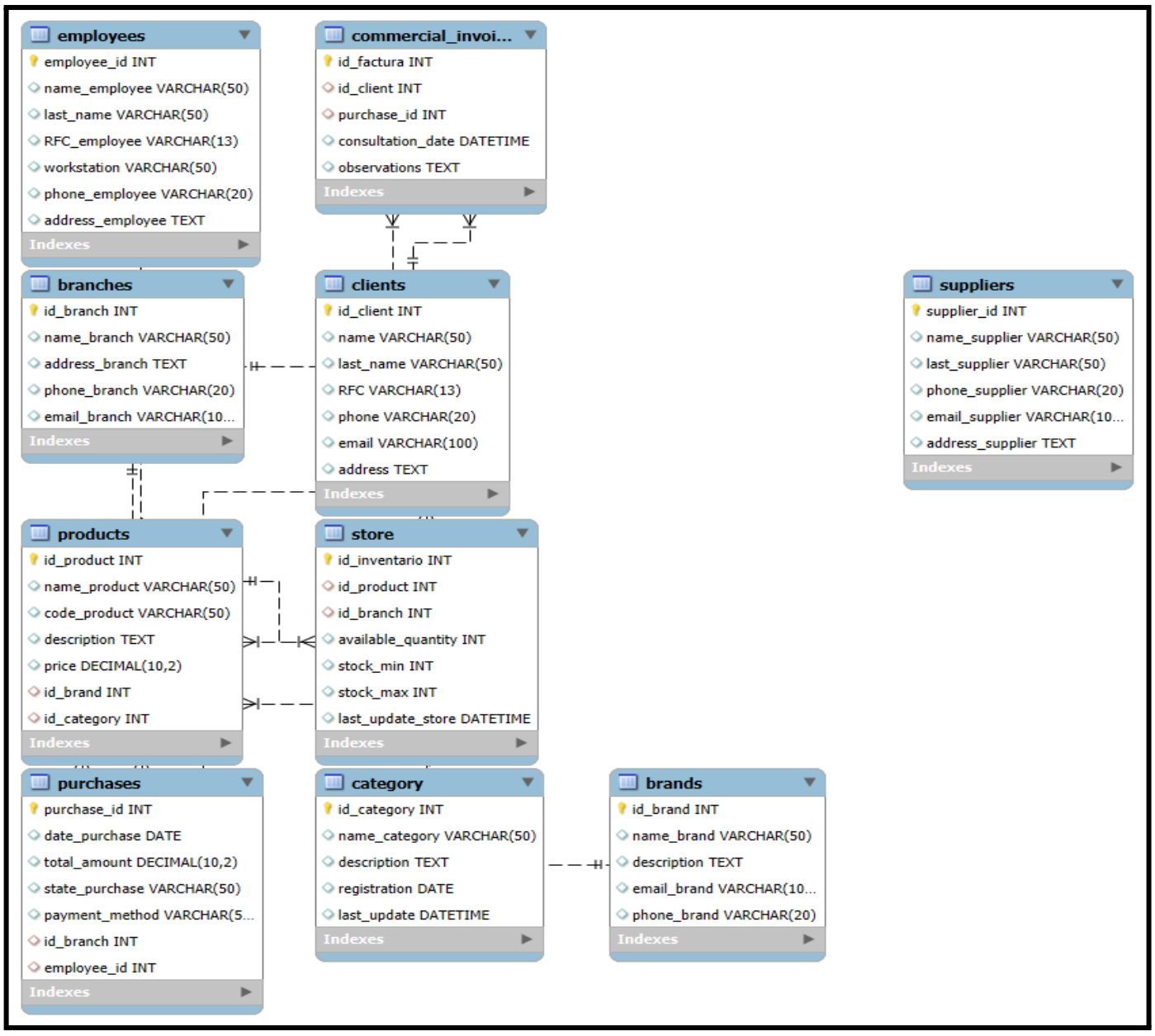
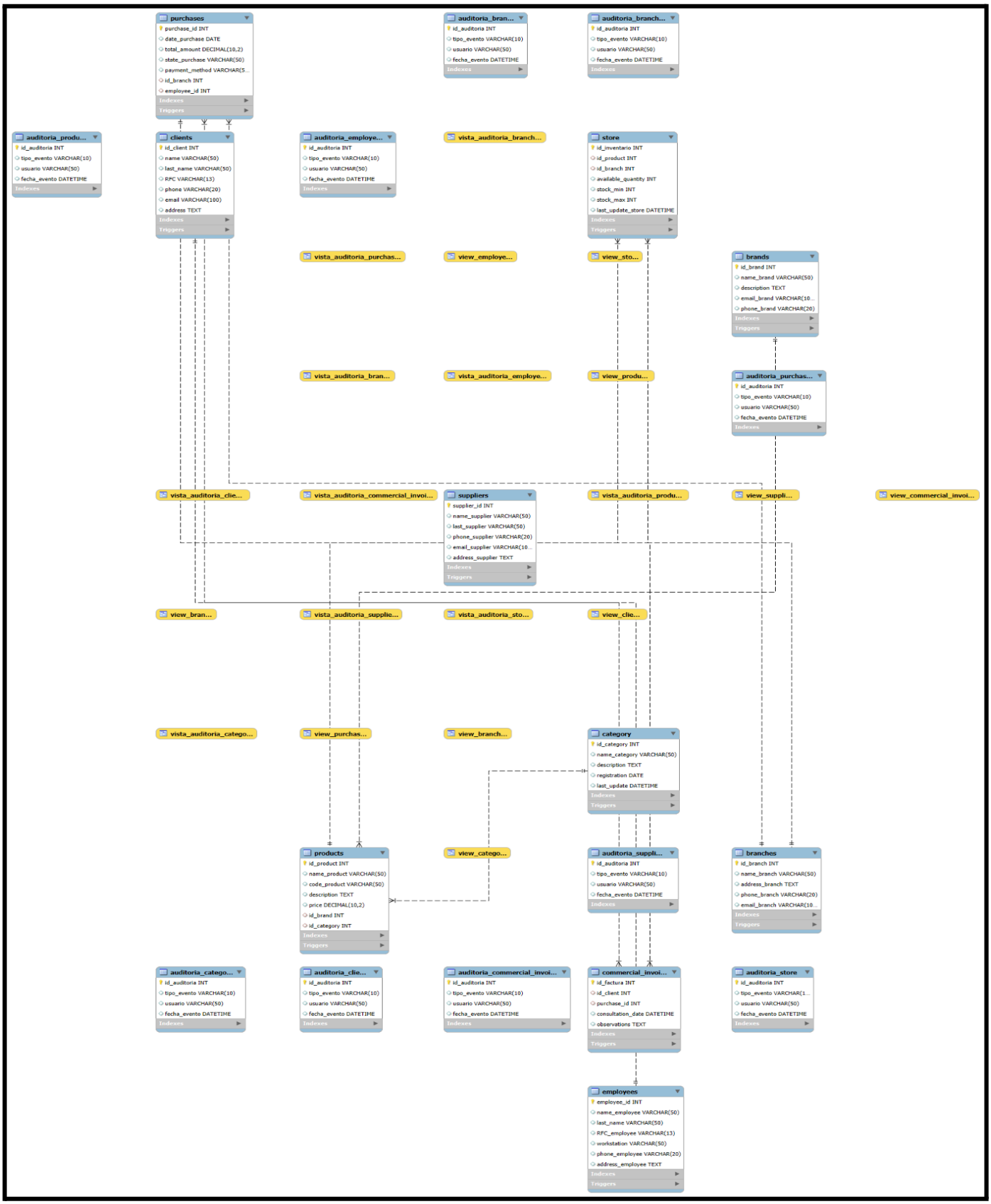


Diagrama Entidad-Relacion (detalles):



Reporte General:

1.- Primero se creó la base de datos y después las tablas que se utilizaran en la estructura del proyecto:

drop database Proyecto_Final_Smoke;

create database if not exists Proyecto_Final_Smoke;

use Proyecto_Final_Smoke;

```
/* En esta seccion se creara la base de datos en caso de que no exista y su uso para empezar a trabajar */
-- drop database Proyecto_Final_Smoke;
create database if not exists Proyecto_Final_Smoke;
use Proyecto_Final_Smoke;
```

-- ========== Estructura Tablas Base de Datos ==========

-- Tabla Clientes

create table Clients (

    id_client int auto_increment primary key,

    name varchar(50),

    last_name varchar(50),

    RFC varchar(13),

    phone varchar(20),

    email varchar(100),

    address text

);

-- Tabla Marcas

create table Brands (

    id_brand int auto_increment primary key,

    name_brand varchar(50),

```
    description text,
    email_brand varchar(100),
    phone_brand varchar(20)
);

-- Tabla Categorías
create table Category (
    id_category int auto_increment primary key,
    name_category varchar(50),
    description text,
    registration date,
    last_update datetime default now()
);

-- Tabla Productos
create table Products (
    id_product int auto_increment primary key,
    name_product varchar(50),
    code_product varchar(50),
    description text,
    price decimal(10,2),
    id_brand int,
    id_category int,
    foreign key (id_brand) references Brands(id_brand),
    foreign key (id_category) references Category(id_category)
);
```

```
-- Tabla Sucursales
create table Branches (
    id_branch int auto_increment primary key,
    name_branch varchar(50),
    address_branch text,
    phone_branch varchar(20),
    email_branch varchar(100)
);

-- Tabla Proveedores
create table Suppliers (
    supplier_id int auto_increment primary key,
    name_supplier varchar(50),
    last_supplier varchar(50),
    phone_supplier varchar(20),
    email_supplier varchar(100),
    address_supplier text
);

-- Tabla Empleados
create table Employees (
    employee_id int auto_increment primary key,
    name_employee varchar(50),
    last_name varchar(50),
    RFC_employee varchar(13),
```

```
    workstation varchar(50),
    phone_employee varchar(20),
    address_employee text
);

-- Tabla Compras
create table Purchases (
    purchase_id int auto_increment primary key,
    date_purchase date,
    total_amount decimal(10,2),
    state_purchase varchar(50),
    payment_method varchar(50),
    id_branch int,
    employee_id int,
    foreign key (id_branch) references Branches(id_branch),
    foreign key (employee_id) references Employees(employee_id)
);

-- Tabla Almacén
create table Store (
    id_inventario int auto_increment primary key,
    id_product int,
    id_branch int,
    available_quantity int,
    stock_min int,
    stock_max int,
```

```
    last_update_store datetime default now(),
    foreign key (id_product) references Products(id_product),
    foreign key (id_branch) references Branches(id_branch)
);

-- Tabla Factura
create table Commercial_invoice (
    id_factura int auto_increment primary key,
    id_client int,
    purchase_id int,
    consultation_date datetime default now(),
    observations text,
    foreign key (id_client) references Clients(id_client),
    foreign key (purchase_id) references Purchases(purchase_id)
);
```

2.- Después se crearon las tablas de Auditoría para los reportes:

```
-- ========================================= Tablas de Auditoria para Reportes
==============================================

-- Auditoría para Clientes
create table auditoria_clients (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);
```

```
-- Auditoría para Marcas
create table auditoria_brands (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Categorías
create table auditoria_category (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Productos
create table auditoria_products (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Sucursales
```

```
create table auditoria_branches (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Proveedores
create table auditoria_suppliers (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Empleados
create table auditoria_employees (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Compras
create table auditoria_purchases (
    id_auditoria int auto_increment primary key,
```

```
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Almacén
create table auditoria_store (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);

-- Auditoría para Facturas
create table auditoria_commercial_invoice (
    id_auditoria int auto_increment primary key,
    tipo_evento varchar(10),
    usuario varchar(50),
    fecha_evento datetime default now()
);
```

3.- Después se crearon las vistas de las tablas de auditoría para su uso:

```
-- ################################## Vistas Tabla auditorias ################################
-- Vista de auditoría para Clientes
create view vista_auditoria_clients as
select * from auditoria_clients;
```

```
-- Vista de auditoría para Marcas
create view vista_auditoria_brands as
select * from auditoria_brands;

-- Vista de auditoría para Categorías
create view vista_auditoria_category as
select * from auditoria_category;

-- Vista de auditoría para Productos
create view vista_auditoria_products as
select * from auditoria_products;

-- Vista de auditoría para Sucursales
create view vista_auditoria_branches as
select * from auditoria_branches;

-- Vista de auditoría para Proveedores
create view vista_auditoria_suppliers as
select * from auditoria_suppliers;

-- Vista de auditoría para Empleados
create view vista_auditoria_employees as
select * from auditoria_employees;

-- Vista de auditoría para Compras
create view vista_auditoria_purchases as
```

```
select * from auditoria_purchases;


-- Vista de auditoría para Almacén

create view vista_auditoria_store as

select * from auditoria_store;


-- Vista de auditoría para Facturas

create view vista_auditoria_commercial_invoice as

select * from auditoria_commercial_invoice;
```

4.- Después se crearon los triggers necesarios para cada una de las tablas de auditoria, en total se crearon 30 triggers, de los cuales 10 son para registrar cuando en las tablas haya inserciones, 10 para registrar cuando se actualicen datos en alguna de las tablas y los últimos 10 son para registrar cuando se elimine un dato de algunas de las tabla:

```
-- ================================== Trigguers para las tablas auditorias
=======================================

-- ========= CLIENTS ========

-- Trigger INSERT corregido

CREATE TRIGGER trg_clients_insert

AFTER INSERT ON clients

FOR EACH ROW

INSERT INTO auditoria_clients(tipo_evento, usuario)

VALUES ('insert', CURRENT_USER());


-- Trigger UPDATE corregido

CREATE TRIGGER trg_clients_update

AFTER UPDATE ON clients

FOR EACH ROW

INSERT INTO auditoria_clients(tipo_evento, usuario)
```

```
VALUES ('update', CURRENT_USER());

-- Trigger DELETE corregido
CREATE TRIGGER trg_clients_delete
AFTER DELETE ON clients
FOR EACH ROW
INSERT INTO auditoria_clients(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

-- Trigger INSERT corregido
CREATE TRIGGER trg_brands_insert
AFTER INSERT ON brands
FOR EACH ROW
INSERT INTO auditoria_brands(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

-- Trigger UPDATE corregido
CREATE TRIGGER trg_brands_update
AFTER UPDATE ON brands
FOR EACH ROW
INSERT INTO auditoria_brands(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

-- Trigger DELETE corregido
CREATE TRIGGER trg_brands_delete
AFTER DELETE ON brands
FOR EACH ROW
INSERT INTO auditoria_brands(tipo_evento, usuario)
```

```
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_category_insert
AFTER INSERT ON category
FOR EACH ROW
INSERT INTO auditoria_category(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_category_update
AFTER UPDATE ON category
FOR EACH ROW
INSERT INTO auditoria_category(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_category_delete
AFTER DELETE ON category
FOR EACH ROW
INSERT INTO auditoria_category(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_products_insert
AFTER INSERT ON products
FOR EACH ROW
INSERT INTO auditoria_products(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_products_update
AFTER UPDATE ON products
```

```
FOR EACH ROW
INSERT INTO auditoria_products(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_products_delete
AFTER DELETE ON products
FOR EACH ROW
INSERT INTO auditoria_products(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_branches_insert
AFTER INSERT ON branches
FOR EACH ROW
INSERT INTO auditoria_branches(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_branches_update
AFTER UPDATE ON branches
FOR EACH ROW
INSERT INTO auditoria_branches(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_branches_delete
AFTER DELETE ON branches
FOR EACH ROW
INSERT INTO auditoria_branches(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());
```

```
CREATE TRIGGER trg_suppliers_insert
AFTER INSERT ON suppliers
FOR EACH ROW
INSERT INTO auditoria_suppliers(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_suppliers_update
AFTER UPDATE ON suppliers
FOR EACH ROW
INSERT INTO auditoria_suppliers(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_suppliers_delete
AFTER DELETE ON suppliers
FOR EACH ROW
INSERT INTO auditoria_suppliers(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_employees_insert
AFTER INSERT ON employees
FOR EACH ROW
INSERT INTO auditoria_employees(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_employees_update
AFTER UPDATE ON employees
FOR EACH ROW
INSERT INTO auditoria_employees(tipo_evento, usuario)
```

```
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_employees_delete
AFTER DELETE ON employees
FOR EACH ROW
INSERT INTO auditoria_employees(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_purchases_insert
AFTER INSERT ON purchases
FOR EACH ROW
INSERT INTO auditoria_purchases(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_purchases_update
AFTER UPDATE ON purchases
FOR EACH ROW
INSERT INTO auditoria_purchases(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_purchases_delete
AFTER DELETE ON purchases
FOR EACH ROW
INSERT INTO auditoria_purchases(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_store_insert
AFTER INSERT ON store
```

```
FOR EACH ROW
INSERT INTO auditoria_store(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_store_update
AFTER UPDATE ON store
FOR EACH ROW
INSERT INTO auditoria_store(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());

CREATE TRIGGER trg_store_delete
AFTER DELETE ON store
FOR EACH ROW
INSERT INTO auditoria_store(tipo_evento, usuario)
VALUES ('delete', CURRENT_USER());

CREATE TRIGGER trg_commercial_invoice_insert
AFTER INSERT ON commercial_invoice
FOR EACH ROW
INSERT INTO auditoria_commercial_invoice(tipo_evento, usuario)
VALUES ('insert', CURRENT_USER());

CREATE TRIGGER trg_commercial_invoice_update
AFTER UPDATE ON commercial_invoice
FOR EACH ROW
INSERT INTO auditoria_commercial_invoice(tipo_evento, usuario)
VALUES ('update', CURRENT_USER());
```

```
CREATE TRIGGER trg_commercial_invoice_delete

AFTER DELETE ON commercial_invoice

FOR EACH ROW

INSERT INTO auditoria_commercial_invoice(tipo_evento, usuario)

VALUES ('delete', CURRENT_USER());
```

5.- Después se crearon 50 stored procedure de los cuales:
a) 10 son para insertar datos en cada tabla
b) 10 son para leer los datos en cada tabla
c) 10 son para leer los datos por id
d) 10 son para actualizar los datos de cada tabla
e) 10 son para eliminar datos de cada tabla

```
-- ====================================== Stored Procedure para Tablas de la Base de Datos
==============================

-- ==== Store Procedure para la tabla clients (insertar)

delimiter //

create procedure insertclient(

in c_name varchar(50),

in c_last_name varchar(50),

in c_rfc varchar(50),

in c_phone varchar(100),

in c_email varchar(100),

in c_address text

)

begin

        insert into Clients (name, last_name, RFC, phone, email, address)

   values(c_name, c_last_name, c_rfc, c_phone, C_email, c_address);

end //

delimiter ;
```

```
-- ==== Store Procedure para la tabla clients (leer)
delimiter //
create procedure get_all_clients()
begin
    select * from clients;
end //
delimiter ;

-- ==== Store Procedure para la tabla clients (leer por id)
delimiter //
create procedure get_client_by_id(
    in c_id int
)
begin
    select * from clients where id_client = c_id;
end //
delimiter ;

-- ==== Store Procedure para la tabla clients (actualizar)
delimiter //
create procedure updateclient(
    in c_id int,
    in c_name varchar(50),
    in c_last_name varchar(50),
    in c_rfc varchar(50),
    in c_phone varchar(100),
    in c_email varchar(100),
    in c_address text
```

```
)
begin
    update clients
    set name = c_name,
        last_name = c_last_name,
        rfc = c_rfc,
        phone = c_phone,
        email = c_email,
        address = c_address
    where id_client = c_id;
end //
delimiter ;

-- ==== Store Procedure para la tabla clients (eliminar)
DELIMITER //
CREATE PROCEDURE deleteclient(
    IN c_id INT
)
BEGIN
    DELETE FROM commercial_invoice -- eliminar facturas asociadas a client
    WHERE id_client = c_id;

    DELETE FROM clients -- eliminar al cliente
    WHERE id_client = c_id;
END //
DELIMITER ;

-- === Stored procedure para tabla branches(insertar)
```

```
delimiter //
create procedure insert_brands(
in b_name varchar(50),
in b_description text,
in b_email_brand varchar(100),
in b_phone_brand varchar(20)
)
begin
        insert into Brands (name_brand, description, email_brand, phone_brand)
   values(b_name, b_description, b_email_brand, b_phone_brand);
end //
delimiter ;

-- ==== Store Procedure para la tabla branches (leer)
delimiter //
create procedure get_all_brands()
begin
   select * from brands;
end //
delimiter ;

-- ==== Store Procedure para la tabla branches (leer por id)
delimiter //
create procedure get_brand_by_id(
   in b_id int
)
begin
   select * from brands
```

```
    where id_brand = b_id;
end //
delimiter ;

-- === Stored procedure para tabla branches(actualizar)
delimiter //
create procedure update_brands(
    in b_id int,
    in b_name varchar(50),
    in b_description text,
    in b_email_brand varchar(100),
    in b_phone_brand varchar(20)
)
begin
    update brands
    set name_brand = b_name,
        description = b_description,
        email_brand = b_email_brand,
        phone_brand = b_phone_brand
    where id_brand = b_id;
end //
delimiter ;

-- eliminar una marca
DELIMITER //
CREATE PROCEDURE delete_brands(
    IN b_id INT
)
```

```
BEGIN
    DELETE FROM store
    WHERE id_product IN (
        SELECT id_product FROM products WHERE id_brand = b_id
    );  -- Eliminar del almacén (store) todos los productos de la marca

    DELETE FROM products -- Eliminar los productos asociados a la marca
    WHERE id_brand = b_id;

    -- Eliminar la marca
    DELETE FROM brands
    WHERE id_brand = b_id;
END //
DELIMITER ;

-- === Stored Procedure para tabla category(insertar)
delimiter //
create procedure insert_category(
in ca_name varchar(50),
in ca_description text,
in ca_registration date
)
begin
            insert into Category (name_category, description, registration)
    values(ca_name, ca_description, ca_registration);
end //
delimiter ;
```

```
-- ==== Store Procedure para la tabla category (leer)
delimiter //
create procedure get_all_categories()
begin
    select * from category;
end //
delimiter ;

-- ==== Store Procedure para la tabla category (leer por id)
delimiter //
create procedure get_category_by_id(
    in ca_id int
)
begin
    select * from category where id_category = ca_id;
end //
delimiter ;

-- === Stored Procedure para tabla category(actualizar)
delimiter //
create procedure update_category(
    in ca_id int,
    in ca_name varchar(50),
    in ca_description text,
    in ca_registration date
)
begin
    update category
```

```
    set name_category = ca_name,
        description = ca_description,
        registration = ca_registration,
        last_update = now()
    where id_category = ca_id;
end //
delimiter ;

-- === Stored Procedure para tabla category(eliminar)
DELIMITER //
CREATE PROCEDURE delete_category(
    IN cat_id INT
)
BEGIN
    -- Eliminar del almacén los productos de esta categoría
    DELETE FROM store
    WHERE id_product IN (
        SELECT id_product FROM products WHERE id_category = cat_id
    );

    -- Eliminar los productos de la categoría
    DELETE FROM products
    WHERE id_category = cat_id;

    -- Eliminar la categoría
    DELETE FROM category
    WHERE id_category = cat_id;
END //
```

```
DELIMITER ;

-- === Stored Procedure para tabla Products(insertar)
delimiter //
create procedure insert_product(
    in p_name varchar(50),
    in p_code varchar(50),
    in p_description text,
    in p_price decimal(10,2),
    in p_id_brand int,
    in p_id_category int
)
begin
    insert into products(name_product, code_product, description, price, id_brand, id_category)
    values(p_name, p_code, p_description, p_price, p_id_brand, p_id_category);
end //
delimiter ;

-- ==== Store Procedure para la tabla products (leer)
delimiter //
create procedure get_all_products()
begin
    select * from products;
end //
delimiter ;

-- ==== Store Procedure para la tabla products (leer por id)
delimiter //
```

```
create procedure get_product_by_id(
    in p_id int
)
begin
    select * from products
    where id_product = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Products(actualizar)
delimiter //
create procedure update_product(
    in p_id int,
    in p_name varchar(50),
    in p_code varchar(50),
    in p_description text,
    in p_price decimal(10,2),
    in p_id_brand int,
    in p_id_category int
)
begin
    update products
    set name_product = p_name,
        code_product = p_code,
        description = p_description,
        price = p_price,
        id_brand = p_id_brand,
        id_category = p_id_category
```

```
    where id_product = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Products(eliminar)
DELIMITER //
CREATE PROCEDURE delete_product(
    IN p_id INT
)
BEGIN
    -- Eliminar del almacén este producto
    DELETE FROM store
    WHERE id_product = p_id;

    -- Eliminar el producto
    DELETE FROM products
    WHERE id_product = p_id;
END //
DELIMITER ;

-- === Stored Procedure para tabla Branches(insertar)
delimiter //
create procedure insert_branch(
    in p_name varchar(50),
    in p_address text,
    in p_phone varchar(20),
    in p_email varchar(100)
)
```

```
begin
    insert into branches(name_branch, address_branch, phone_branch, email_branch)
    values(p_name, p_address, p_phone, p_email);
end //
delimiter ;

-- ==== Store Procedure para la tabla Branches(leer)
delimiter //
create procedure get_all_branches()
begin
    select * from branches;
end //
delimiter ;

-- ==== Store Procedure para la tabla Branches(leer por id)
delimiter //
create procedure get_branch_by_id(
    in p_id int
)
begin
    select * from branches
    where id_branch = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Branches(actualizar)
delimiter //
create procedure update_branch(
```

```
    in p_id int,
    in p_name varchar(50),
    in p_address text,
    in p_phone varchar(20),
    in p_email varchar(100)
)
begin
    update branches
    set name_branch = p_name,
        address_branch = p_address,
        phone_branch = p_phone,
        email_branch = p_email
    where id_branch = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Branches(eliminar)
DELIMITER //
CREATE PROCEDURE delete_branch(
    IN b_id INT
)
BEGIN
    -- Eliminar facturas ligadas a compras de esta sucursal
    DELETE FROM commercial_invoice
    WHERE purchase_id IN (
        SELECT purchase_id FROM purchases WHERE id_branch = b_id
    );
```

```
    -- Eliminar compras asociadas a esta sucursal
    DELETE FROM purchases
    WHERE id_branch = b_id;

    -- Eliminar productos en el almacén de esta sucursal
    DELETE FROM store
    WHERE id_branch = b_id;

    -- Eliminar la sucursal
    DELETE FROM branches
    WHERE id_branch = b_id;
END //
DELIMITER ;

-- === Stored Procedure para tabla Suppliers(insertar)
delimiter //
create procedure insert_supplier(
    in p_name varchar(50),
    in p_last varchar(50),
    in p_phone varchar(20),
    in p_email varchar(100),
    in p_address text
)
begin
    insert into suppliers(name_supplier, last_supplier, phone_supplier, email_supplier, address_supplier)
    values(p_name, p_last, p_phone, p_email, p_address);
end //
delimiter ;
```

```
-- ==== Store Procedure para la tabla Suppliers(leer)
delimiter //
create procedure get_all_suppliers()
begin
    select * from suppliers;
end //
delimiter ;

-- ==== Store Procedure para la tabla Suppliers(leer por id)
delimiter //
create procedure get_supplier_by_id(
    in p_id int
)
begin
    select * from suppliers
    where supplier_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Suppliers(actualizar)
delimiter //
create procedure update_supplier(
    in p_id int,
    in p_name varchar(50),
    in p_last varchar(50),
    in p_phone varchar(20),
    in p_email varchar(100),
```

```
    in p_address text
)
begin
    update suppliers
    set name_supplier = p_name,
        last_supplier = p_last,
        phone_supplier = p_phone,
        email_supplier = p_email,
        address_supplier = p_address
    where supplier_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Suppliers(eliminar)
delimiter //
create procedure delete_supplier(
    in p_id int
)
begin
    delete from suppliers
    where supplier_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Employees(insertar)
delimiter //
create procedure insert_employee(
    in p_name varchar(50),
```

```
    in p_last varchar(50),
    in p_rfc varchar(13),
    in p_workstation varchar(50),
    in p_phone varchar(20),
    in p_address text
)
begin
    insert into employees(name_employee, last_name, rfc_employee, workstation, phone_employee, address_employee)
    values(p_name, p_last, p_rfc, p_workstation, p_phone, p_address);
end //
delimiter ;

-- ==== Store Procedure para la tabla Employees(leer)
delimiter //
create procedure get_all_employees()
begin
    select * from employees;
end //
delimiter ;

-- ==== Store Procedure para la tabla Employees(leer por id)
delimiter //
create procedure get_employee_by_id(
    in p_id int
)
begin
    select * from employees
    where employee_id = p_id;
```

```
end //
delimiter ;

-- === Stored Procedure para tabla Employees(actualizar)
delimiter //
create procedure update_employee(
    in p_id int,
    in p_name varchar(50),
    in p_last varchar(50),
    in p_rfc varchar(13),
    in p_workstation varchar(50),
    in p_phone varchar(20),
    in p_address text
)
begin
    update employees
    set name_employee = p_name,
        last_name = p_last,
        rfc_employee = p_rfc,
        workstation = p_workstation,
        phone_employee = p_phone,
        address_employee = p_address
    where employee_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Employees(eliminar)
DELIMITER //
```

```
CREATE PROCEDURE delete_employee(
    IN emp_id INT
)
BEGIN
    DELETE FROM commercial_invoice -- Eliminar facturas asociadas a compras hechas por este empleado
    WHERE purchase_id IN (
        SELECT purchase_id FROM purchases WHERE employee_id = emp_id
    );

    DELETE FROM purchases -- Eliminar compras hechas por este empleado
    WHERE employee_id = emp_id;

    DELETE FROM employees  -- Eliminar al empleado
    WHERE employee_id = emp_id;
END //
DELIMITER ;

-- === Stored Procedure para tabla Purchases(insertar)
delimiter //
create procedure insert_purchase(
    in p_date date,
    in p_total decimal(10,2),
    in p_state varchar(50),
    in p_payment varchar(50),
    in p_branch int,
    in p_employee int
)
begin
```

```
    insert into purchases(date_purchase, total_amount, state_purchase, payment_method, id_branch, employee_id)
    values(p_date, p_total, p_state, p_payment, p_branch, p_employee);
end //
delimiter ;

-- ==== Store Procedure para la tabla Purchases(leer)
delimiter //
create procedure get_all_purchases()
begin
    select * from purchases;
end //
delimiter ;

-- ==== Store Procedure para la tabla Purchases(leer por id)
delimiter //
create procedure get_purchase_by_id(
    in p_id int
)
begin
    select * from purchases
    where purchase_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Purchases(actualizar)
delimiter //
create procedure update_purchase(
    in p_id int,
```

```
    in p_date date,
    in p_total decimal(10,2),
    in p_state varchar(50),
    in p_payment varchar(50),
    in p_branch int,
    in p_employee int
)
begin
    update purchases
    set date_purchase = p_date,
        total_amount = p_total,
        state_purchase = p_state,
        payment_method = p_payment,
        id_branch = p_branch,
        employee_id = p_employee
    where purchase_id = p_id;
end //
delimiter ;

-- === Stored Procedure para tabla Purchases(eliminar)
DELIMITER //
CREATE PROCEDURE delete_purchase(
    IN pur_id INT
)
BEGIN
    -- Eliminar facturas asociadas a esta compra
    DELETE FROM commercial_invoice
    WHERE purchase_id = pur_id;
```

```
    -- Eliminar la compra
    DELETE FROM purchases
    WHERE purchase_id = pur_id;
END //
DELIMITER ;

-- === Stored Procedure para tabla Store(insertar)
delimiter //
create procedure insert_store(
    in p_product int,
    in p_branch int,
    in p_quantity int,
    in p_min int,
    in p_max int
)
begin
    insert into store(id_product, id_branch, available_quantity, stock_min, stock_max)
    values(p_product, p_branch, p_quantity, p_min, p_max);
end //
delimiter ;

-- ==== Store Procedure para la tabla Store(leer)
delimiter //
create procedure get_all_store()
begin
    select * from store;
end //
```

```
delimiter ;

-- ==== Store Procedure para la tabla Store(leer por id)
delimiter //
create procedure get_store_by_id(
    in p_id_inventario int
)
begin
    select * from store
    where id_inventario = p_id_inventario;
end //
delimiter ;

-- === Stored Procedure para tabla Store(actualizar)
delimiter //
create procedure update_store(
    in p_id_inventario int,
    in p_product int,
    in p_branch int,
    in p_quantity int,
    in p_min int,
    in p_max int
)
begin
    update store
    set id_product = p_product,
        id_branch = p_branch,
        available_quantity = p_quantity,
```

```
        stock_min = p_min,
        stock_max = p_max,
        last_update_store = now()
    where id_inventario = p_id_inventario;
end //
delimiter ;

-- === Stored Procedure para tabla Store(eliminar)
delimiter //
create procedure delete_store(
    in p_id_inventario int
)
begin
    delete from store
    where id_inventario = p_id_inventario;
end //
delimiter ;

-- === Stored Procedure para tabla Commercial_invoice(insertar)
delimiter //
create procedure insert_invoice(
    in p_client int,
    in p_purchase int,
    in p_observations text
)
begin
    insert into commercial_invoice(id_client, purchase_id, observations)
    values(p_client, p_purchase, p_observations);
```

```
end //
delimiter ;

-- ==== Store Procedure para la tabla Commercial_invoice(leer)
delimiter //
create procedure get_all_invoices()
begin
   select * from commercial_invoice;
end //
delimiter ;

-- ==== Store Procedure para la tabla Commercial_invoice(leer por id)
delimiter //
create procedure get_invoice_by_id(
   in p_id_factura int
)
begin
   select * from commercial_invoice
   where id_factura = p_id_factura;
end //
delimiter ;

-- === Stored Procedure para tabla Commercial_invoice(actualizar)
delimiter //
create procedure update_invoice(
   in p_id_factura int,
   in p_client int,
   in p_purchase int,
```

```
    in p_observations text
)
begin
    update commercial_invoice
    set id_client = p_client,
        purchase_id = p_purchase,
        observations = p_observations,
        consultation_date = now()
    where id_factura = p_id_factura;
end //
delimiter ;

-- === Stored Procedure para tabla Commercial_invoice(eliminar)
delimiter //
create procedure delete_invoice(
    in p_id_factura int
)
begin
    delete from commercial_invoice
    where id_factura = p_id_factura;
end //
delimiter ;
```

6.- Después se crearon las vistas de cada tabla de la base de datos para su consulta:

```
-- ********************************************** Vistas de cada tabla Base de Datos
**********************************************

-- ****** Vista Clients

create view view_clients as

select * from clients;


-- ****** Vista Brands

create view view_brands as

select * from brands;


-- ****** Vista Category

create view view_category as

select * from category;


-- ****** Vista Products

create view view_products as

select * from products;


-- ****** Vista Branches

create view view_branches as

select * from branches;


-- ****** Vista Suppliers

create view view_suppliers as

select * from suppliers;


-- ****** Vista Employees

create view view_employees as
```

```
select * from employees;


-- ****** Vista Purchases

create view view_purchases as

select * from purchases;


-- ****** Vista Store

create view view_store as

select * from store;


-- ****** Vista Commercial_invoice

create view view_commercial_invoice as

select * from commercial_invoice;
```

7.- Luego se crearon las llamadas de los stored procedure para obtener los datos de cada una de las tablas:

```
call get_all_brands();

call get_all_categories();

call get_all_products();

call get_all_branches();

call get_all_suppliers();

call get_all_employees();

call get_all_purchases();

call get_all_store();

call get_all_invoices();
```

8. Despues se crearon las llamadas para insertar, leer todos los datos, leer un dato por id, actualizar los datos y eliminar un dato por id: (por cada una de las tablas)

-- == tabla clients
==

-- insertar:

CALL insertclient('Ana', 'Pérez', 'PEAA850101ABC', '5544332211', 'ana.perez@gmail.com', 'Calle Reforma 123, CDMX');

CALL insertclient('Luis', 'Ramírez', 'RALU920203DEF', '5512345678', 'luis.ramirez@hotmail.com', 'Av. Juárez 456, CDMX');

CALL insertclient('María', 'González', 'GOMA900707GHI', '5523456789', 'maria.g@gmail.com', 'Insurgentes Sur 789, CDMX');

CALL insertclient('Carlos', 'Sánchez', 'SACA880808JKL', '5534567890', 'carlos.s@hotmail.com', 'Eje Central 321, CDMX');

CALL insertclient('Laura', 'Martínez', 'MALA950909MNO', '5545678901', 'laura.martinez@yahoo.com', 'Av. Universidad 654, CDMX');

CALL insertclient('Fernando', 'López', 'LOFE930303PQR', '5556789012', 'fer.lopez@gmail.com', 'Col. Roma 111, CDMX');

CALL insertclient('Patricia', 'Hernández', 'HEPA861212STU', '5567890123', 'paty.h@gmail.com', 'Col. Del Valle 222, CDMX');

CALL insertclient('Jorge', 'Torres', 'TOJO970101VWX', '5578901234', 'jorge.torres@gmail.com', 'Tacuba 333, CDMX');

CALL insertclient('Sofía', 'Flores', 'FLSO990202YZA', '5589012345', 'sofia.flores@gmail.com', 'Av. Patriotismo 444, CDMX');

CALL insertclient('Diego', 'Reyes', 'REDI870505BCD', '5590123456', 'diego.reyes@gmail.com', 'Mixcoac 555, CDMX');

CALL insertclient('Mónica', 'Ortiz', 'ORMO910606EFG', '5543211234', 'moni.ortiz@hotmail.com', 'Tlalpan 666, CDMX');

CALL insertclient('Eduardo', 'Castillo', 'CAED880707HIJ', '5532124321', 'edu.castillo@gmail.com', 'Napoles 777, CDMX');

CALL insertclient('Isabel', 'Ríos', 'RIIS950808KLM', '5521032103', 'isa.rios@gmail.com', 'Av. Revolución 888, CDMX');

CALL insertclient('Manuel', 'Morales', 'MOMA920909NOP', '5512348765', 'manuel.morales@gmail.com', 'Viaducto 999, CDMX');

CALL insertclient('Gabriela', 'Cruz', 'CRGA880101QRS', '5509876543', 'gabriela.cruz@gmail.com', 'Roma Norte 101, CDMX');

CALL insertclient('Alberto', 'Méndez', 'MEAL910202TUV', '5510987654', 'alberto.mendez@gmail.com', 'Narvarte 202, CDMX');

CALL insertclient('Elena', 'Salas', 'SAEL920303WXY', '5521098765', 'elena.salas@gmail.com', 'Del Carmen 303, CDMX');

CALL insertclient('Ricardo', 'Vega', 'VERI900404ZAB', '5532109876', 'ricardo.vega@gmail.com', 'Centro 404, CDMX');

CALL insertclient('Daniela', 'Navarro', 'NADA931212CDE', '5543210987', 'daniela.n@gmail.com', 'Portales 505, CDMX');

```
CALL insertclient('Alejandro', 'Ibarra', 'IBAL890505FGH', '5554321098', 'ale.ibarra@gmail.com', 'Azcapotzalco 606, CDMX');

CALL insertclient('Valeria', 'Delgado', 'DEVA950707IJK', '5565432109', 'val.delgado@gmail.com', 'Centro Histórico 707, CDMX');

CALL insertclient('Roberto', 'Campos', 'CARO870808LMN', '5576543210', 'roberto.campos@gmail.com', 'Doctores 808, CDMX');

CALL insertclient('Andrea', 'Silva', 'SIAN960909OPQ', '5587654321', 'andrea.silva@gmail.com', 'Escandón 909, CDMX');

CALL insertclient('Héctor', 'Cortés', 'COHE920101RST', '5598765432', 'hector.cortes@gmail.com', 'Santa María 010, CDMX');

CALL insertclient('Lucía', 'Molina', 'MOLU870202UVW', '5501234567', 'lucia.molina@gmail.com', 'San Ángel 111, CDMX');

CALL insertclient('Ángel', 'Serrano', 'SEAN940303XYZ', '5512345670', 'angel.serrano@gmail.com', 'Buenavista 212, CDMX');

CALL insertclient('Verónica', 'Padilla', 'PAVE930404ABC', '5523456781', 'vero.padilla@gmail.com', 'Industrial 313, CDMX');

CALL insertclient('Ernesto', 'Fuentes', 'FUER910505DEF', '5534567892', 'ernesto.fuentes@gmail.com', 'Chapultepec 414, CDMX');

CALL insertclient('Regina', 'Carrillo', 'CARE890606GHI', '5545678903', 'regina.carrillo@gmail.com', 'Tlatelolco 515, CDMX');

CALL insertclient('Julián', 'Medina', 'MEJU960707JKL', '5556789014', 'julian.medina@gmail.com', 'Xochimilco 616, CDMX');

CALL insertclient('Teresa', 'Luna', 'LUTE850808MNO', '5567890125', 'teresa.luna@gmail.com', 'Magdalena 717, CDMX');

CALL insertclient('Marco', 'León', 'LEMA970909PQR', '5578901236', 'marco.leon@gmail.com', 'Centro 818, CDMX');

CALL insertclient('Rebeca', 'Aguilar', 'AGRE860101STU', '5589012347', 'rebeca.aguilar@gmail.com', 'Pedregal 919, CDMX');

CALL insertclient('Óscar', 'Durán', 'DUOS880202VWX', '5590123458', 'oscar.duran@gmail.com', 'Roma Sur 020, CDMX');

CALL insertclient('Natalia', 'Peña', 'PENA900303YZA', '5509876541', 'natalia.pena@gmail.com', 'Juárez 121, CDMX');

CALL insertclient('Iván', 'Solís', 'SOIV920404BCD', '5510987652', 'ivan.solis@gmail.com', 'Del Valle 222, CDMX');

CALL insertclient('Brenda', 'Valle', 'VABR930505EFG', '5521098763', 'brenda.valle@gmail.com', 'Mixcoac 323, CDMX');

CALL insertclient('Armando', 'Escobar', 'ESAR910606HIJ', '5532109874', 'armando.escobar@gmail.com', 'Tacubaya 424, CDMX');

CALL insertclient('Flor', 'Lara', 'LAFL950707KLM', '5543210985', 'flor.lara@gmail.com', 'Santa Úrsula 525, CDMX');

CALL insertclient('Esteban', 'Rivas', 'RIES890808NOP', '5554321096', 'esteban.rivas@gmail.com', 'Copilco 626, CDMX');

-- leer todos los registros
```

```
call get_all_clients();

-- leer por id

call get_client_by_id(1);

-- actualizar

CALL updateclient(1, 'Ana Luisa', 'Pérez Ramirez', 'PEAA850101ABC', '5544332212', 'ana.maria@gmail.com', 'Av. Juárez 456, CDMX');

CALL updateclient(2, 'Luis Fernando', 'Ramírez Gómez', 'RALU920203DEF', '5512345679', 'luis.fernando@gmail.com', 'Av. Reforma 789, CDMX');

CALL updateclient(3, 'María Elena', 'González Ruiz', 'GOMA900707GHI', '5523456790', 'maria.elena@gmail.com', 'Calle Londres 321, CDMX');

CALL updateclient(4, 'Carlos Alberto', 'Sánchez Díaz', 'SACA880808JKL', '5534567891', 'carlos.a.sanchez@gmail.com', 'Av. Insurgentes 101, CDMX');

CALL updateclient(5, 'Laura Beatriz', 'Martínez Ríos', 'MALA950909MNO', '5545678902', 'laura.b.martinez@gmail.com', 'Col. Roma Sur 654, CDMX');




-- ================================================ tabla branches
==================================================

-- insertar

CALL insert_brands('PSYCODELIA', 'ropa y artículos con estilo psicodélico', 'contacto@psycodelia.com', '5551000001');

CALL insert_brands('STIIIZY', 'productos de vapeo y cartuchos premium', 'soporte@stiiizy.com', '5551000002');

CALL insert_brands('Elf Bar', 'dispositivos electrónicos desechables para vapeo', 'info@elfbar.com', '5551000003');

CALL insert_brands('GEEK BAR', 'tecnología de vapeo compacta y moderna', 'ventas@geekbar.com', '5551000004');

CALL insert_brands('LOST MARY', 'vapeadores con sabores exóticos y diseño moderno', 'contact@lostmary.com', '5551000005');

CALL insert_brands('Vaporesso', 'dispositivos avanzados para vapeo', 'info@vaporesso.com', '5551000006');

CALL insert_brands('V-Play', 'productos de entretenimiento y tecnología', 'contacto@vplay.com', '5551000007');

CALL insert_brands('Nimmbox', 'gadgets y dispositivos electrónicos portátiles', 'ventas@nimmbox.com', '5551000008');

CALL insert_brands('GRINDER', 'accesorios para preparación de hierbas y especias', 'info@grinder.com', '5551000009');

CALL insert_brands('BACKWOODS', 'puros y productos de tabaco de estilo natural', 'soporte@backwoods.com', '5551000010');

-- leer todos los registros
```

```
call get_all_brands();

-- leer por id

call get_brand_by_id(2);

-- actualizar

CALL update_brands(2, 'PSYCODELIA', 'marca de ropa alternativa y psicodélica', 'contacto@psycodelia.com',
'5551222233');

CALL update_brands(3, 'STIIIZY', 'líder en cartuchos de vapeo premium', 'soporte@stiiizy.com', '5551333344');

CALL update_brands(4, 'Elf Bar', 'vapeadores desechables con gran variedad de sabores', 'info@elfbar.com', '5551444455');

CALL update_brands(5, 'GEEK BAR', 'vapeadores compactos de última generación', 'ventas@geekbar.com', '5551555566');

CALL update_brands(6, 'LOST MARY', 'marca reconocida por su diseño y sabores únicos en vapeo',
'contact@lostmary.com', '5551666677');



-- ================================================ tabla category
==================================================

-- insertar

CALL insert_category('Ropa y accesorios', 'Prendas de vestir, accesorios y estilo personal', '2025-01-10');

CALL insert_category('Vapeadores y cartuchos', 'Dispositivos de vapeo recargables y sus cartuchos', '2025-01-10');

CALL insert_category('Vapeadores desechables', 'Dispositivos de vapeo de un solo uso', '2025-01-10');

CALL insert_category('Dispositivos de vapeo', 'Equipos electrónicos avanzados para vapeo', '2025-01-10');

CALL insert_category('Electrónica / Entretenimiento digital', 'Gadgets, consolas, y productos digitales', '2025-01-10');

CALL insert_category('Dispositivos electrónicos / Gadgets', 'Tecnología portátil, accesorios y herramientas inteligentes',
'2025-01-10');

CALL insert_category('Accesorios para hierbas / Molinillos', 'Grinders y utensilios para preparación de hierbas',
'2025-01-10');

CALL insert_category('Productos de tabaco / Puros', 'Puros, hojas, y productos derivados del tabaco', '2025-01-10');

-- leer todos los registros

call get_all_categories();

-- leer por id

call get_category_by_id(2);

-- actualizar
```

```
CALL update_category(2, 'Vapeadores y cartuchos', 'dispositivos recargables y cartuchos compatibles', '2025-01-12');

CALL update_category(3, 'Electrónica', 'gadgets, dispositivos inteligentes y artículos digitales', '2025-01-12');

CALL update_category(4, 'Accesorios para hierbas', 'molinillos y utensilios para preparación de hierbas', '2025-01-12');



-- ========================================== tabla products
==================================================

-- insertar

CALL insert_product('HHC GUMMIE BLUEBERRY PUNCH', 'P001', 'Gomitas con HHC sabor blueberry punch, 100mg - 50/Pack', 599.00, 1, 2);

CALL insert_product('STIIIZY HHC GUMMIES POG', 'P002', '15pc HHC Gummies POG Sativa 825MG - 10 PACK', 749.00, 2, 2);

CALL insert_product('STIIIZY DELTA 8 GUMMIES MANGO TANGO', 'P003', '15pc DELTA 8 Mango Tango Sativa 1500MG - 10 PACK', 799.00, 2, 2);

CALL insert_product('Elf Bar TE 5000 Blueberry Ice', 'P004', 'Disposable 5% (10/pack 130ml) Blueberry Ice', 950.00, 3, 3);

CALL insert_product('Elf Bar TE 5000 Grape Apple Ice', 'P005', 'Disposable 5% (10/pack 130ml) Grape Apple Ice', 950.00, 3, 3);

CALL insert_product('Elf Bar BC 10000 DOUBLE MANGO', 'P006', 'Disposable 5% (5/pack) - Double Mango', 999.00, 3, 3);

CALL insert_product('Elf Bar BC 10000 MIAMI MINT', 'P007', 'Disposable 5% (5/pack) - Miami Mint', 999.00, 3, 3);

CALL insert_product('GEEK BAR MELOSO MAX TROPICAL', 'P008', '9000 DISPOSABLE 5% - Tropical Rainbow Blast', 980.00, 4, 3);

CALL insert_product('LOST MARY ORANGE PINE NANA MANGO', 'P009', 'MO 10000 Disposable 5% - Orange Pineapple Nana Mango', 1020.00, 5, 3);

CALL insert_product('Vaporesso XROS 3 Mini Kit', 'P010', 'Kit Phantom Gold edición mini', 890.00, 6, 4);

CALL insert_product('V-Play 20K Merry Berry', 'P011', 'Disposable 25mL - Merry Berry 50mg (5/pack)', 950.00, 7, 5);

CALL insert_product('Nimmbox Blue Razz', 'P012', '10000 PUFFS DISPOSABLE 50mg - Blue Razz', 920.00, 8, 6);

CALL insert_product('BONGA VIDRIO WJ09', 'P013', 'Pipa de vidrio en proceso de búsqueda', 450.00, 8, 6);

CALL insert_product('GRINDER Zinc 55mm-3', 'P014', 'Grinder metálico Zinc 55mm, 6 PACK', 680.00, 9, 7);

CALL insert_product('JAM MONSTER BLUEBERRY', 'P015', 'Líquido 0MG 100ML sabor Blueberry', 260.00, 2, 2);

CALL insert_product('ORGNX SALT ZERO DEGREES', 'P016', 'Sal de nicotina 30ML 35MG', 240.00, 2, 2);

CALL insert_product('BACKWOODS Smoking Set Bag', 'P017', 'Estuche de accesorios BACKWOODS (TZN0008)', 330.00, 10, 8);
```

```
CALL insert_product('RAW CLASSIC CONES 20PK', 'P018', '98mm/20mm Conos clásicos 12/display', 190.00, 10, 8);

CALL insert_product('PSYCODELIA BANDANA ALIEN', 'P019', 'Bandana 55x55cm estilo alien - negra', 110.00, 1, 1);

CALL insert_product('PSYCODELIA GRAMERA FIRST EDITION', 'P020', 'Báscula digital edición especial', 280.00, 1, 1);

-- leer todos los registros

call get_all_products();

-- leer por id

call get_product_by_id(2);

-- actualizar

CALL update_product(1, 'HHC GUMMIE BLUEBERRY PUNCH 50PK', 'P001A', 'Gomitas HHC sabor blueberry punch 100mg, paquete de 50', 609.00, 1, 2);

CALL update_product(2, 'STIIIZY GUMMIES POG Sativa 825MG', 'P002A', 'Paquete de 10 con 15 gomitas HHC sabor POG, Sativa', 759.00, 2, 2);

CALL update_product(3, 'STIIIZY DELTA 8 GUMMIES Mango Tango', 'P003A', 'Gomitas Delta 8 Mango Tango, 15 piezas Sativa 1500mg', 809.00, 2, 2);

CALL update_product(4, 'Elf Bar TE5000 Blueberry Ice 10pk', 'P004A', 'Vape desechable 5%, Blueberry Ice, 10 por paquete', 960.00, 3, 3);

CALL update_product(5, 'Elf Bar TE5000 Grape Apple Ice', 'P005A', 'Vape desechable 5%, Grape Apple Ice, 10 por paquete', 960.00, 3, 3);



-- ========================================= tabla branches(sucursales)
=================================================

-- insertar

CALL insert_branch('sucursal Aguacaliente', 'Av. Aguacaliente 101', '5551000001', 'aguacaliente@empresa.com');

CALL insert_branch('sucursal Alameda', 'Calle Alameda 202', '5551000002', 'alameda@empresa.com');

CALL insert_branch('sucursal Fuentes', 'Blvd. Las Fuentes 303', '5551000003', 'fuentes@empresa.com');

CALL insert_branch('sucursal Centro 1', 'Av. Reforma 123', '5551000004', 'centro1@empresa.com');

CALL insert_branch('sucursal Centro 2', 'Av. Juárez 456', '5551000005', 'centro2@empresa.com');

CALL insert_branch('sucursal Soler', 'Calle Soler 789', '5551000006', 'soler@empresa.com');

CALL insert_branch('sucursal Sendero', 'Blvd. Sendero 147', '5551000007', 'sendero@empresa.com');

CALL insert_branch('sucursal Tecate', 'Av. Hidalgo 369', '5551000008', 'tecate@empresa.com');

CALL insert_branch('sucursal Del Valle', 'Insurgentes Sur 4321, CDMX', '5551000009', 'delvalle@empresa.com');
```

```
CALL insert_branch('sucursal Polanco', 'Av. Presidente Masaryk 22, CDMX', '5551000010', 'polanco@empresa.com');

CALL insert_branch('sucursal Condesa', 'Av. Tamaulipas 150, CDMX', '5551000011', 'condesa@empresa.com');

-- leer todos los registros

call get_all_branches();

-- leer por id

call get_branch_by_id(2);

-- actualizar

CALL update_branch(2, 'sucursal Aguacaliente renovada', 'Av. Aguacaliente 202', '5552000001',
'aguacaliente202@empresa.com');

CALL update_branch(3, 'sucursal Alameda Express', 'Calle Alameda 333', '5552000002', 'alameda.express@empresa.com');

CALL update_branch(4, 'sucursal Fuentes Norte', 'Blvd. Las Fuentes 404', '5552000003', 'fuentesnorte@empresa.com');


-- ========================================== tabla suppliers (proveedores)
====================================

-- insertar

CALL insert_supplier('Psycodelia México S.A. de C.V.', 'Soporte', '5559000001', 'contacto@psycodelia.com', 'Av. Reforma
101, CDMX');

CALL insert_supplier('Stiiizy Distribución Global', 'Atención Cliente', '5559000002', 'ventas@stiiizy.com', 'Calle Vape 202,
Los Ángeles, CA');

CALL insert_supplier('Elf Bar Supply Co.', 'Distribuidor', '5559000003', 'info@elfbar.com', 'Av. Asia 321, Shenzhen, China');

CALL insert_supplier('Geek Vape Distribution', 'Logística', '5559000004', 'distribucion@geekbar.com', 'Calle Tecnología 44,
CDMX');

CALL insert_supplier('Lost Mary International', 'Pedidos', '5559000005', 'contact@lostmary.com', 'Av. Vapor 7, Miami, FL');

CALL insert_supplier('Vaporesso Tech Corp', 'Ventas', '5559000006', 'info@vaporesso.com', 'Calle Shenzhen 88, China');

CALL insert_supplier('V-Play Electronics MX', 'Atención MX', '5559000007', 'contacto@vplay.com', 'Insurgentes Sur 500,
CDMX');

CALL insert_supplier('Nimmbox Vapes Inc.', 'Soporte', '5559000008', 'ventas@nimmbox.com', 'Av. Vapor 99, Nueva York,
NY');

CALL insert_supplier('Grinder Solutions S.A.', 'Ventas', '5559000009', 'info@grinder.com', 'Calle Molino 123, CDMX');

CALL insert_supplier('Backwoods Tobacco Distributors', 'Distribución', '5559000010', 'soporte@backwoods.com', 'Ruta del
Tabaco 11, Miami, FL');
```

```
CALL insert_supplier('RAW Papers International', 'Servicio', '5559000011', 'contact@rawpapers.com', 'Av. Hemp 20, Barcelona, España');

CALL insert_supplier('Monster Vapes Supply', 'Atención', '5559000012', 'info@jammonster.com', 'Av. Monster Vape 55, Chicago, IL');

CALL insert_supplier('Headshop Central Distributors', 'Logística', '5559000013', 'contact@headshopcentral.com', 'Blvd. Alternativo 33, Tijuana, BC');

CALL insert_supplier('Psycodelia México Accesorios', 'Accesorios', '5559000014', 'accesorios@psycodelia.com', 'Calle Gnomo 42, CDMX');

-- leer todos los registros

call get_all_suppliers();

-- leer por id

call get_supplier_by_id(2);

-- actualizar

CALL update_supplier(1, 'Psycodelia', 'Soporte', '5559000001', 'contacto@psycodelia.com', 'Av. Reforma 101, CDMX');

CALL update_supplier(2, 'Stiiizy', 'Distribución', '5559000002', 'ventas@stiiizy.com', 'Calle Vape 202, Los Ángeles, CA');

CALL update_supplier(3, 'Elf', 'Distribuidor', '5559000003', 'info@elfbar.com', 'Av. Asia 321, Shenzhen, China');


-- ============================================= tabla empleados
=================================================

-- insertar

CALL insert_employee('Luis', 'Martínez', 'MAML900303XYZ', 'Almacenista', '5551237890', 'Calle Central 45');

CALL insert_employee('María', 'López', 'LOPM850707DEF', 'Supervisora', '5559876543', 'Boulevard del Sol 321');

CALL insert_employee('José', 'Ramírez', 'RAJJ880909GHI', 'Vendedor', '5556543210', 'Colonia Jardines 12');

CALL insert_employee('Carmen', 'Fernández', 'FECA920202JKL', 'Cajera', '5553217890', 'Av. Universidad 300');

CALL insert_employee('Diego', 'Hernández', 'HEDI931010MNO', 'Repartidor', '5558765432', 'Zona Centro 88');

CALL insert_employee('Lucía', 'Mendoza', 'MELU870505PQR', 'Encargada de tienda', '5554321987', 'Calle Palmas 200');

CALL insert_employee('Carlos', 'Núñez', 'NUCA940808STU', 'Almacenista', '5552983746', 'Calle Sur 56');

CALL insert_employee('Jessica', 'Torres', 'TOJE910606VWX', 'Cajera', '5556789432', 'Av. Norte 120');

CALL insert_employee('Pedro', 'Santos', 'SAPD950404YZA', 'Vendedor', '5557346281', 'Calle Robles 67');

-- leer todos los registros
```

```
call get_all_employees();

-- leer por id

call get_employee_by_id(2);

-- actualizar

CALL update_employee(1, 'Lucía', 'Mendoza', 'MEAL900101XYZ', 'Gerente', '5559988776', 'Calle Principal 789');

CALL update_employee(2, 'Luis', 'Martínez', 'MAML900303XYZ', 'Supervisor de Almacén', '5551237899', 'Calle Reforma
222');

CALL update_employee(3, 'María', 'López', 'LOPM850707DEF', 'Jefa de Caja', '5559876500', 'Avenida del Trabajo 110');


-- ========================================= tabla purchases
============================================================

-- insertar

CALL insert_purchase('2025-05-20', 1500.00, 'completado', 'efectivo', 1, 1);

CALL insert_purchase('2025-05-21', 2300.50, 'pendiente', 'transferencia', 2, 3);

CALL insert_purchase('2025-05-22', 1850.75, 'completado', 'tarjeta', 3, 2);

CALL insert_purchase('2025-05-23', 2120.00, 'cancelado', 'efectivo', 4, 4);

CALL insert_purchase('2025-05-24', 950.00, 'completado', 'efectivo', 5, 5);

CALL insert_purchase('2025-05-25', 3200.00, 'completado', 'transferencia', 6, 6);

CALL insert_purchase('2025-05-26', 1680.90, 'pendiente', 'tarjeta', 7, 7);

CALL insert_purchase('2025-05-27', 740.00, 'completado', 'efectivo', 8, 8);

CALL insert_purchase('2025-05-28', 2890.60, 'completado', 'transferencia', 9, 9);

CALL insert_purchase('2025-05-29', 1999.99, 'pendiente', 'efectivo', 2, 3);

CALL insert_purchase('2025-05-30', 1700.00, 'completado', 'tarjeta', 1, 2);

CALL insert_purchase('2025-05-31', 2500.40, 'cancelado', 'efectivo', 2, 3);

CALL insert_purchase('2025-06-01', 1350.80, 'completado', 'transferencia', 3, 4);

CALL insert_purchase('2025-06-02', 1100.00, 'pendiente', 'efectivo', 4, 5);

CALL insert_purchase('2025-06-03', 2050.00, 'completado', 'tarjeta', 5, 6);

CALL insert_purchase('2025-06-04', 1450.55, 'completado', 'transferencia', 6, 7);

CALL insert_purchase('2025-06-05', 760.20, 'cancelado', 'efectivo', 7, 8);
```

```
CALL insert_purchase('2025-06-06', 1899.99, 'completado', 'efectivo', 8, 9);

CALL insert_purchase('2025-06-07', 2240.00, 'pendiente', 'transferencia', 5, 8);

CALL insert_purchase('2025-06-08', 950.00, 'completado', 'tarjeta', 1, 1);

-- leer todos los registros

call get_all_purchases();

-- leer por id

call get_purchase_by_id(2);

-- actualizar

CALL update_purchase(4, '2025-05-22', 2100.00, 'procesando', 'transferencia', 2, 3);

CALL update_purchase(5, '2025-05-23', 1750.50, 'completado', 'efectivo', 3, 4);

CALL update_purchase(6, '2025-05-24', 1980.75, 'pendiente', 'tarjeta', 4, 5);

CALL update_purchase(7, '2025-05-25', 2250.00, 'cancelado', 'transferencia', 5, 6);

CALL update_purchase(8, '2025-05-26', 1890.00, 'procesando', 'efectivo', 6, 1);


-- ================================================== tabla store
==================================================

-- insertar

CALL insert_store(1, 1, 100, 10, 200);

CALL insert_store(2, 2, 150, 20, 250);

CALL insert_store(3, 3, 80, 15, 180);

CALL insert_store(4, 4, 120, 10, 220);

CALL insert_store(5, 5, 90, 25, 190);

CALL insert_store(6, 6, 200, 30, 300);

CALL insert_store(7, 7, 110, 20, 210);

CALL insert_store(8, 8, 95, 15, 195);

CALL insert_store(9, 9, 130, 25, 230);

CALL insert_store(10, 10, 140, 20, 240);

-- leer todos los registros
```

```
call get_all_store();

-- leer por id

call get_store_by_id(2);

-- actualizar

CALL update_store(1, 1, 1, 80, 10, 200);

CALL update_store(2, 2, 2, 150, 20, 250);

CALL update_store(3, 3, 3, 90, 15, 190);


-- ======================================= tabla commercial_invoice(factura)
=========================================

-- insertar

CALL insert_invoice(1, 1, 'entregado sin observaciones');

CALL insert_invoice(2, 2, 'entregado con retraso');

CALL insert_invoice(3, 3, 'entregado con daños menores');

CALL insert_invoice(4, 4, 'pendiente de entrega');

CALL insert_invoice(5, 5, 'entregado sin observaciones');

CALL insert_invoice(6, 6, 'entregado con observaciones');

CALL insert_invoice(7, 7, 'entregado sin observaciones');

CALL insert_invoice(8, 8, 'entregado con retraso');

CALL insert_invoice(9, 9, 'entregado sin observaciones');

CALL insert_invoice(8, 8, 'entregado con daños menores');

CALL insert_invoice(6, 1, 'pendiente de entrega');

CALL insert_invoice(1, 2, 'entregado sin observaciones');

CALL insert_invoice(3, 3, 'entregado con observaciones');

CALL insert_invoice(6, 4, 'entregado sin observaciones');

CALL insert_invoice(7, 5, 'entregado con retraso');

CALL insert_invoice(2, 6, 'pendiente de entrega');

CALL insert_invoice(1, 7, 'entregado sin observaciones');
```

```
CALL insert_invoice(3, 8, 'entregado con observaciones');

CALL insert_invoice(5, 9, 'entregado sin observaciones');

CALL insert_invoice(20, 10, 'entregado con retraso');

-- leer todos los registros

call get_all_invoices();

-- leer por id

call get_invoice_by_id(2);

-- actualizar

CALL update_invoice(1, 1, 1, 'entregado con observaciones');

CALL update_invoice(2, 2, 2, 'entregado sin observaciones');

CALL update_invoice(3, 3, 3, 'pendiente de entrega');

CALL update_invoice(4, 4, 4, 'entregado con retraso');

CALL update_invoice(5, 5, 5, 'entregado sin observaciones');

-- ========================================== Eliminar
==========================================

-- eliminar clients

call deleteclient(5);

-- eliminar marcas

call delete_brands(1);

-- eliminar categoria

call delete_category(5);

-- eliminar producto

call delete_product(9);

-- eliminar branches

call delete_branch(1);

-- eliminar suppliers

call delete_supplier(4);

-- eliminar empleado
```

```
call delete_employee(5);

-- eliminar purchase

call delete_purchase(2);

-- eliminar store

call delete_store(3);

-- eliminar factura

call delete_invoice(1);
```

9. Luego se crearon las llamadas para cada una de las tablas de auditoría para su uso:

```
-- ============================================== llamadas a tabla auditoria
==============================================

-- Clientes

SELECT * FROM vista_auditoria_clients;


-- Marcas

SELECT * FROM vista_auditoria_brands;


-- Categorías

SELECT * FROM vista_auditoria_category;


-- Productos

SELECT * FROM vista_auditoria_products;


-- Sucursales

SELECT * FROM vista_auditoria_branches;


-- Proveedores

SELECT * FROM vista_auditoria_suppliers;
```

```
-- Empleados

SELECT * FROM vista_auditoria_employees;


-- Compras

SELECT * FROM vista_auditoria_purchases;


-- Almacén

SELECT * FROM vista_auditoria_store;


-- Facturas

SELECT * FROM vista_auditoria_commercial_invoice;
```

10. Luego se realizaron los select para cada una de las vistas de las tablas que conforman la base de datos:

```
-- >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>  Llamar vistas Base de Datos
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

-- >>>> ver Clients

select * from view_clients;

-- >>>> ver Brands

select * from view_brands;

-- >>>> ver Category

select * from view_category;

-- >>>> ver Products

select * from view_products;

-- >>>> ver Branches

select * from view_branches;

-- >>>> ver Suppliers

select * from view_suppliers;
```

```
-- >>>> ver Employees

select * from view_employees;

-- >>>> ver Purchases

select * from view_purchases;

-- >>>> ver Store

select * from view_store;

-- >>>> ver Commercial_invoice

select * from view_commercial_invoice;
```

Conclusión:

Este proyecto final de Base de Datos Avanzadas permitió aplicar de forma práctica los conocimientos adquiridos durante el curso. Se diseñaron tablas relacionales, procedimientos almacenados, triggers y vistas que mejoraron la gestión, seguridad y eficiencia de los datos. Las consultas realizadas sobre las vistas facilitaron la obtención de información clave, demostrando la utilidad de una base de datos bien estructurada. En conjunto, el trabajo fortaleció nuestras habilidades técnicas y nos preparó para enfrentar retos reales en el ámbito profesional.