RETO BASE DE DATOS, TIENDA VIRTUAL DON PEPE (EJERCICIO C)

JESÚS MIGUEL MOLINA MENDOZA

SOFKA U

TRANING LEAGUE AUTOMATIZACIÓN DE PRUEBAS

2023

Tabla de contenido

1.0 Indicar que ejercicio fue asignado.

Me fue asignado el ejercicio tienda virtual don pepe (Ejercicio C).

2.0 Realizar el modelo E-R

Entidades:

Cliente: cedula, nombre, dirección, teléfono, email, password, zona.

Producto: id, nombre, marca, origen, contenido, fotografía, stock.

Categoría: id, nombre, condiciones de almacenamiento, observaciones.

Proveedor: id, nombre, teléfono.

Pedido: id, datos de pago (número de tarjeta, fecha de caducidad), total, dirección del pedido, fecha pedido.

Domiciliario: id, numero de matrícula, nombre, zona.

Relaciones:

Realiza: Cliente 1:N Pedido, porque un cliente puede realizar muchos pedidos, pero un pedido solo puede ser realizado por cliente.

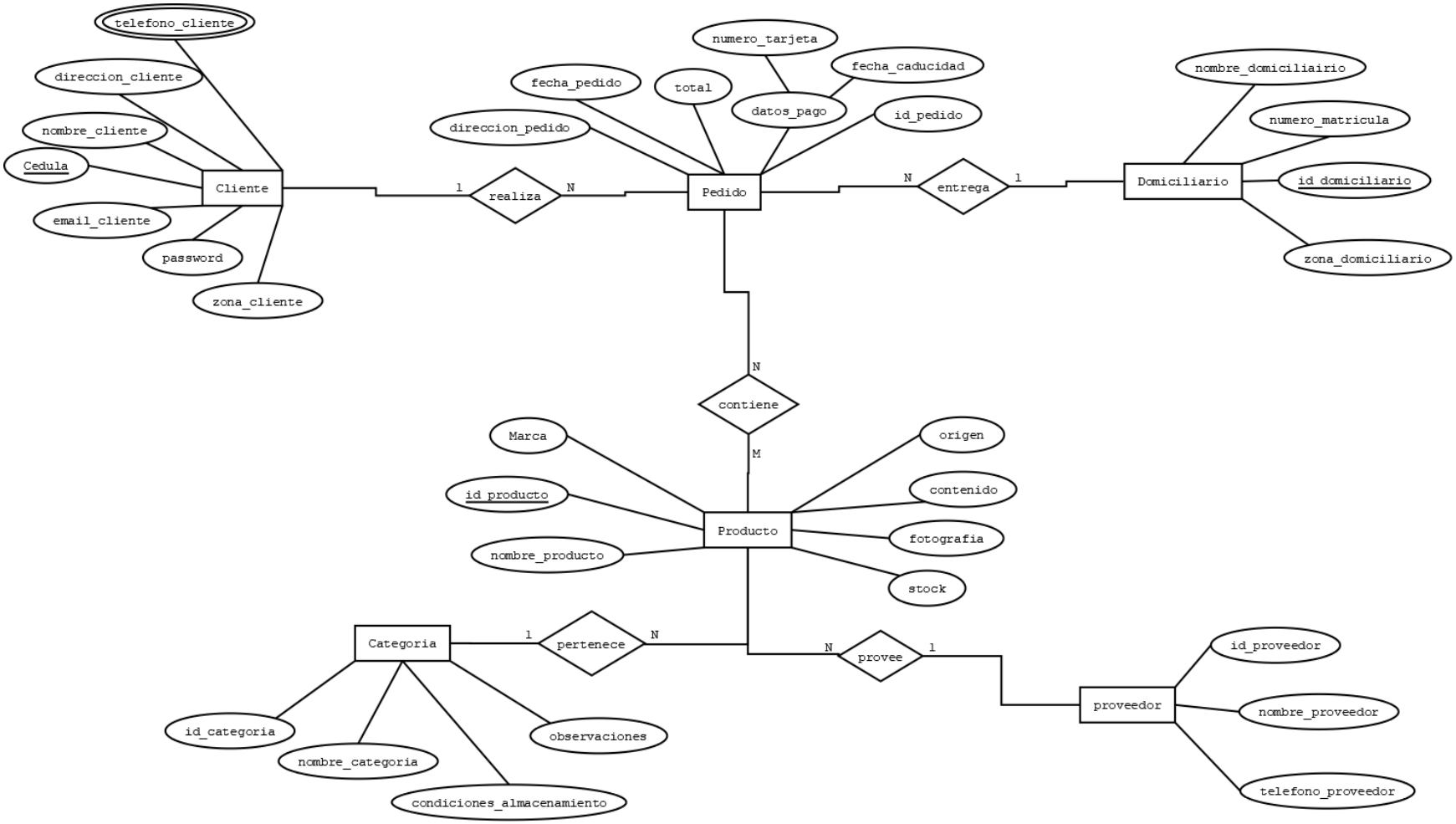
Entrega: pedido N:1 domiciliario, porque un pedido puede ser entregado por un domiciliario, pero un domiciliario puede entregar muchos pedidos.

Contiene: cesta compra N:M producto, porque en una cesta de compra puede contener muchos productos y un producto puede estar en muchas cestas de compra.

Pertenece: categoría 1:N producto, porque a una categoría pueden pertenecer muchos productos, pero un producto solo puede pertenecer a una categoría.

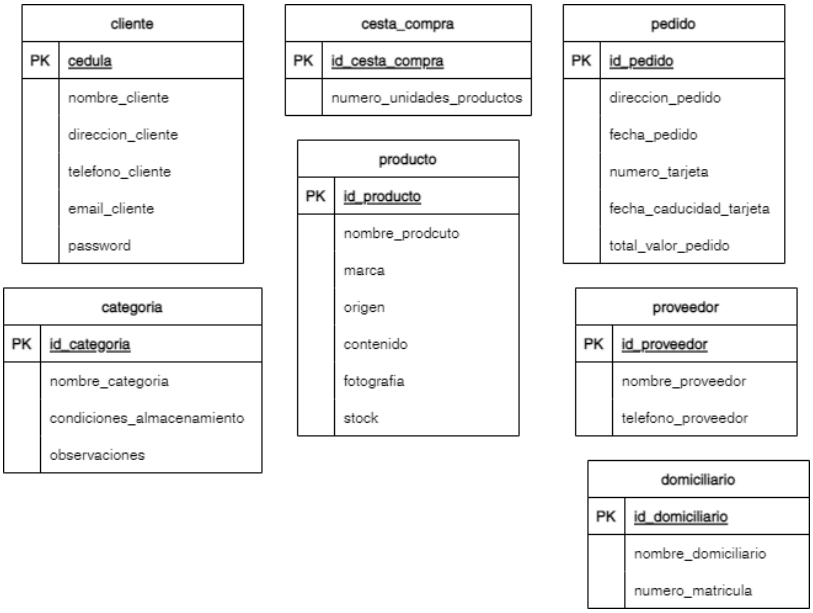
Provee: proveedor 1:N producto, porque un proveedor puede provisionar muchos productos pero un producto solo puede ser provisionado por un proveedor.

A partir de esto se crea el siguiente diagrama:

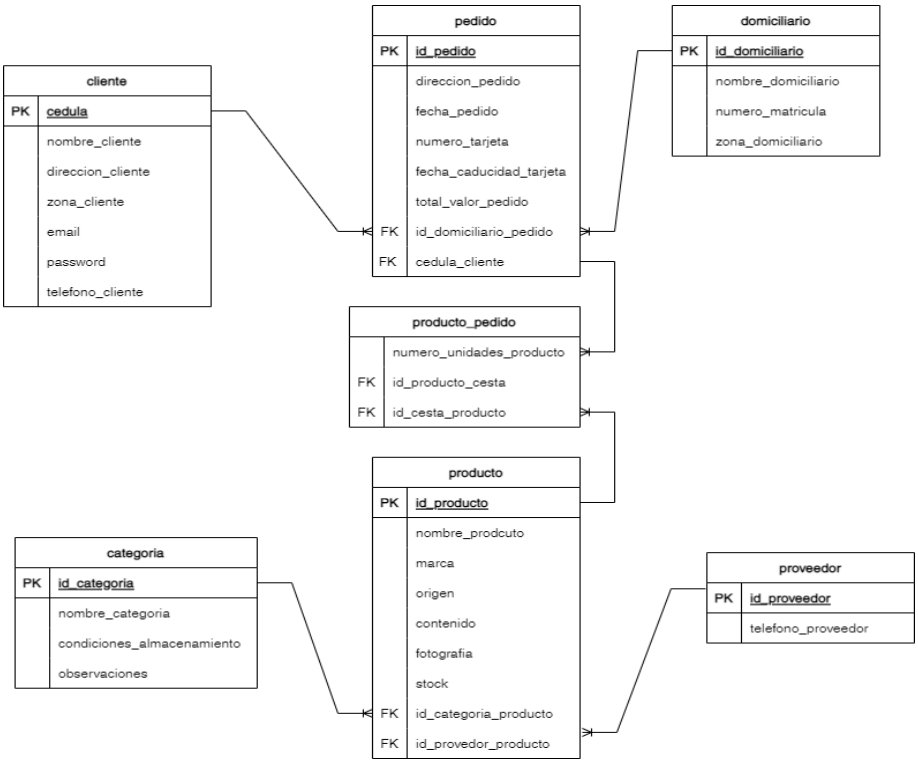


3.0 Realizar el modelo relacional

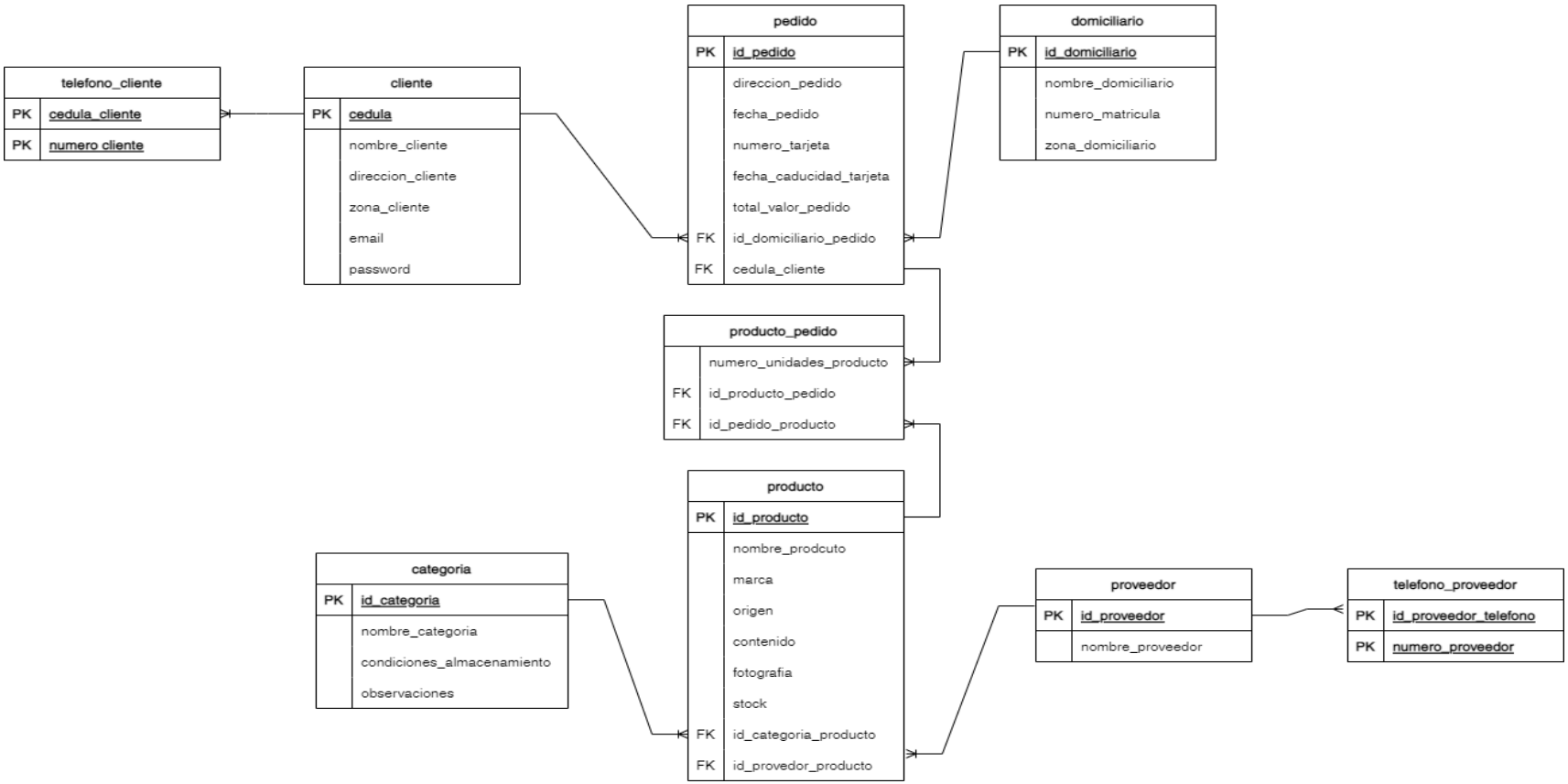
Se convierten las entidades en tablas.



Se hacen modifican las tablas según sus relaciones y se crean nuevas tablas para relaciones muchos a muchos.



Después de normalizar el diagrama quedaría de la siguiente forma.



de esta forma la tabla cumple todas las formas normales, ya que dejaron de existir atributos multivalor, es decir ahora todos son atómicos, todas las tablas poseen una llave primaria única, cada tabla depende de su llave primaria y no existe una dependencia transitiva entre atributos que no son clave

4.0 Escribir con sentencias SQL toda la definición de la base de datos.

Usamos la sentencia CREATE DATABASE + el nombre de la base de datos (en este caso tienda) para crear la base de datos. Adicional a eso, usamos la sentencia USE para seleccionar una base de datos especifica. en este caso sería la base de datos tienda.

```
1     CREATE DATABASE tienda;
2
3     USE tienda;
```

Para crear una tabla usamos la sentencia CREATE TABLE + el nombre de la tabla y dentro de paréntesis van a ir el nombre de los atributos con el tipo de dato, además que se le debe colocar PRIMARY KEY al dato que sea la llave primaria y NOT NULL a los datos que deban ser obligatoriamente llenados.

```
6  CREATE TABLE cliente (
7  cedula VARCHAR(15) NOT NULL PRIMARY KEY,
8  nombre_cliente VARCHAR(50) NOT NULL,
9  direccion_cliente VARCHAR(50) NOT NULL,
10 zona_cliente VARCHAR(15) NOT NULL,
11 email VARCHAR(30) NOT NULL,
12 contraseña VARCHAR(20) NOT NULL
13 );
```

Para las tablas que tengan relaciones de uno a muchos, en la tabla que está del lado de muchos se agrega un nuevo atributo con el nombre de la llave primaria de la tabla que esta de lado de uno y se agrega la sentencia FOREIGN KEY con el nombre de la llave foránea y REFERENCES con el nombre de la tabla donde vienen esa llave y entre paréntesis el nombre de la llave principal.

```
69 CREATE TABLE pedido(
70 id_pedido VARCHAR(15) NOT NULL PRIMARY KEY,
71 direccion_pedido VARCHAR(50) NOT NULL,
72 fecha_pedido DATETIME DEFAULT current_timestamp,
73 numero_tarjeta VARCHAR(20) NOT NULL,
74 fecha_caducidad_tarjeta VARCHAR(5) NOT NULL,
75 total_valor VARCHAR(10) NOT NULL,
76 cedula_cliente VARCHAR(15) NOT NULL,
77 id_domiciliario_pedido VARCHAR(15) NOT NULL,
78 FOREIGN KEY(cedula_cliente) REFERENCES cliente(cedula),
79 FOREIGN KEY(id_domiciliario_pedido) REFERENCES domiciliario(id_domiciliario)
80 );
```

Para las tablas que tienen relaciones muchos a muchos se crea una nueva tabla con el nombre de las dos tablas, y se crean dos atributos con el nombre de las llaves primarias de cada tabla. además, se asignan las llaves foráneas referenciando las tablas que se relacionan y las llaves primarias de cada tabla.

```
83  CREATE TABLE producto_pedido(
84  id_producto_pedido VARCHAR(15) NOT NULL,
85  id_pedido_producto VARCHAR(15) NOT NULL,
86  Cantidad INT NOT NULL CHECK (Cantidad >= 1),
87  FOREIGN KEY(id_producto_pedido) REFERENCES producto(id_producto),
88  FOREIGN KEY(id_pedido_producto) REFERENCES pedido(id_pedido)
89  );
```

Para las tablas que tienen atributos multivalor (por ejemplo, el teléfono del cliente) se crea una nueva tabla con dos atributos que serán llaves principales, uno que va a ser la llave principal de la tabla cliente y otro va a ser el número del cliente. Y se coloca como llave foránea el atributo del id del técnico.

```
15      -- tabla telefono cliente
16  CREATE TABLE telefono_cliente (
17    cedula_cliente VARCHAR(15) NOT NULL,
18    numero_cliente VARCHAR(15) NOT NULL,
19    PRIMARY KEY (cedula_cliente, numero_cliente),
20    FOREIGN KEY (cedula_cliente) REFERENCES cliente (cedula)
21  );
```

Finalmente, el script queda así:

```sql
CREATE DATABASE tienda;

USE tienda;

-- tabla cliente
CREATE TABLE cliente (
cedula VARCHAR(15) NOT NULL PRIMARY KEY,
nombre_cliente VARCHAR(50) NOT NULL,
direccion_cliente VARCHAR(50) NOT NULL,
zona_cliente VARCHAR(15) NOT NULL,
email VARCHAR(30) NOT NULL,
contraseña VARCHAR(20) NOT NULL
);

-- tabla telefono cliente
CREATE TABLE telefono_cliente (
  cedula_cliente VARCHAR(15) NOT NULL,
  numero_cliente VARCHAR(15) NOT NULL,
  PRIMARY KEY (cedula_cliente, numero_cliente),
  FOREIGN KEY (cedula_cliente) REFERENCES cliente (cedula)
);

-- tabla proveedor
CREATE TABLE proveedor(
id_proveedor  VARCHAR(15) NOT NULL PRIMARY KEY,
nombre_proveedor VARCHAR(50) NOT NULL
);

-- tabla telefono proveedor
CREATE TABLE telefono_proveedor(
id_proveedor_telefono VARCHAR(15) NOT NULL,
numero_proveedor VARCHAR(15) NOT NULL,
PRIMARY KEY(id_proveedor_telefono, numero_proveedor),
FOREIGN KEY (id_proveedor_telefono) REFERENCES proveedor(id_proveedor)
);
```

```sql
-- tabla categoria
CREATE TABLE categoria(
id_categoria VARCHAR(15) NOT NULL PRIMARY KEY,
nombre_categoria VARCHAR(50) NOT NULL,
condiciones_almacenaimiento VARCHAR(15) NOT NULL,
observaciones VARCHAR(15) NOT NULL
);

-- tabla producto
CREATE TABLE producto(
id_producto VARCHAR(15) NOT NULL PRIMARY KEY,
nombre_producto VARCHAR(50) NOT NULL,
marca VARCHAR(50) NOT NULL,
origen VARCHAR(20) NOT NULL,
contenido VARCHAR(20) NOT NULL,
fotografia VARCHAR(20) NOT NULL,
stock INT NOT NULL,
id_categoria_producto VARCHAR(15) NOT NULL,
id_proveedor_producto VARCHAR(15) NOT NULL,
FOREIGN KEY(id_categoria_producto) REFERENCES categoria(id_categoria),
FOREIGN KEY(id_proveedor_producto) REFERENCES proveedor(id_proveedor)
);

-- tabla domiciliario
CREATE TABLE domiciliario(
id_domiciliario VARCHAR(15) NOT NULL PRIMARY KEY,
nombre_domiciliario VARCHAR(50) NOT NULL,
numero_matricula VARCHAR(20) NOT NULL,
zona_domiciliario VARCHAR(15) NOT NULL
);


-- tabla pedido
CREATE TABLE pedido(
id_pedido VARCHAR(15) NOT NULL PRIMARY KEY,
direccion_pedido VARCHAR(50) NOT NULL,
fecha_pedido DATETIME DEFAULT current_timestamp,
numero_tarjeta VARCHAR(20) NOT NULL,
fecha_caducidad_tarjeta VARCHAR(5) NOT NULL,
total_valor VARCHAR(10) NOT NULL,
cedula_cliente VARCHAR(15) NOT NULL,
id_domiciliario_pedido VARCHAR(15) NOT NULL,
FOREIGN KEY(cedula_cliente) REFERENCES cliente(cedula),
FOREIGN KEY(id_domiciliario_pedido) REFERENCES domiciliario(id_domiciliario)
);
```

```sql
-- tabla producto_pedido
CREATE TABLE producto_pedido(
id_producto_pedido VARCHAR(15) NOT NULL,
id_pedido_producto VARCHAR(15) NOT NULL,
Cantidad INT NOT NULL CHECK (Cantidad >= 1),
FOREIGN KEY(id_producto_pedido) REFERENCES producto(id_producto),
FOREIGN KEY(id_pedido_producto) REFERENCES pedido(id_pedido)
);
```

5.0 Escribir consultas que me permitan ver la información de cada tabla o de varias tablas (10).

Consulta pata ver el listado de productos con su categoría y proveedor:

Para esta consulta se hizo un SELECT a producto.nombre_producto, categoria.nombre_categoria, proveedor.nombre_proveedor que son los campos que se quieren mostrar en la consulta, además dos INNER JOIN desde producto, uno a la tabla categoría y otro a proveedor.

```sql
SELECT producto.nombre_producto, categoria.nombre_categoria, proveedor.nombre_proveedor
FROM producto
INNER JOIN categoria ON categoria.id_categoria = producto.id_categoria_producto
INNER JOIN proveedor ON proveedor.id_proveedor = producto.id_proveedor_producto;
```

nombre_producto	nombre_categoria	nombre_proveedor
Producto A	Categoría A	Proveedor A
Producto B	Categoría B	Proveedor B

Consulta parar ver los clientes con su número de teléfono:

Para esta consulta se hizo un SELECT a cliente.nombre_cliente AS nombre, telefono_cliente.numero_cliente AS teléfono que son los campos que se quieren mostrar en la consulta, además se hizo un INNER JOIN desde cliente a la tabla teléfono_cliente.

```sql
-- 2 consulta para ver el cliente con su numero de telefono
SELECT cliente.nombre_cliente AS nombre, telefono_cliente.numero_cliente AS telefono
FROM cliente
INNER JOIN telefono_cliente ON telefono_cliente.cedula_cliente = cliente.cedula;
```

nombre	telefono
Juan Pérez	555-1234
María Gómez	555-5678

consulta para ver e proveedores con su número de teléfono:

Para esta consulta se hizo un SELECT a proveedor.nombre_proveedor, telefono_proveedor.numero_proveedor que son los campos que se quieren mostrar en la consulta, además se hizo un INNER JOIN desde proveedor a la tabla telefono_proveedor.

```
-- 3 consulta para ver e proveedores con su numero de telefono
SELECT proveedor.nombre_proveedor, telefono_proveedor.numero_proveedor
FROM proveedor
INNER JOIN telefono_proveedor ON telefono_proveedor.id_proveedor_telefono = proveedor.id_proveedor;
```

nombre_proveedor	numero_proveedor
Proveedor A	555-1234
Proveedor B	555-5678

Consulta para ver el pedido con los productos y las cantidades:

Para esta consulta se hizo un SELECT a pedido.id_pedido, y se uso un GROUP_CONCAT para concatenar producto.nombre_producto, producto_pedido.Cantidad, y se hicieron dos INNER JOIN desde producto_pedido hacia pedido y producto.

```
SELECT pedido.id_pedido, GROUP_CONCAT( producto.nombre_producto, ' (', producto_pedido.Cantidad, ') ') as productos_cantidad
FROM producto_pedido
INNER JOIN pedido ON producto_pedido.id_pedido_producto = pedido.id_pedido
INNER JOIN producto ON producto_pedido.id_producto_pedido = producto.id_producto
GROUP BY pedido.id_pedido;
```

id_pedido	productos_cantidad
PED001	Producto A (2) ,Producto B (5)
PED002	Producto B (3)

Consulta para ver los clientes con pedidos mayores a 60000

Para esta consulta se hizo un SELECT a cliente.nombre_cliente, cliente.cedula, pedido.total_valor y un INNER JOIN desde cliente a pedido luego se hizo un WHERE para pedir que solo mostrar los pedidos con un valor mayor a 60000.

```
-- 5 consulta para ver los clientes con pedidos mayores a 60000
SELECT cliente.nombre_cliente, cliente.cedula, pedido.total_valor
FROM cliente
INNER JOIN pedido ON cliente.cedula = pedido.cedula_cliente
WHERE pedido.total_valor > 60000;
```

nombre_cliente	cedula	total_valor
María Gómez	987654321	75000

Consulta para ver domiciliarios en la zona 2:

Se usa un SELECT * para pedir que traiga todos los campos de la tabla domiciliario y un WHERE para pedir que solo traiga los que estén en la zona 2

```
SELECT *
FROM domiciliario
WHERE zona_domiciliario = 'Zona 2';
```

id_domiciliario	nombre_domiciliario	numero_matricula	zona_domiciliario
dom002	Domiciliario 2	M0002	Zona 2

Consulta para ver los domiciliarios que hay en cada zona de cada cliente

Para esta consulta se hizo un SELECT a liente.nombre_cliente, cliente.zona_cliente, domiciliario.nombre_domiciliario, domiciliario.zona_domiciliario que son los campos que se quieren mostrar en la consulta, además se hizo un INNER JOIN desde cliente a la tabla telefono_domiciliario

```
156 •  SELECT cliente.nombre_cliente, cliente.zona_cliente, domiciliario.nombre_domiciliario, domiciliario.zona_domiciliario
157    FROM cliente
158    INNER JOIN domiciliario ON cliente.zona_cliente = domiciliario.zona_domiciliario;
```

nombre_cliente	zona_cliente	nombre_domiciliario	zona_domiciliario
Juan Pérez	Zona 1	Domiciliario 1	Zona 1
María Gómez	Zona 2	Domiciliario 2	Zona 2

Consulta para ver los productos con sus stock:

Para esto se hizo un SELECT producto.nombre_producto, producto.stock desde la tabla producto

```
SELECT producto.nombre_producto, producto.stock
FROM producto;
```

nombre_producto	stock
Producto A	50
Producto B	100

Consulta para ver las direcciones id del pedido y el domiciliario que lo entrega:

Para esto se hizo un SELECT a domiciliario.nombre_domiciliario, pedido.id_pedido desde la tabla domiciliario y INNER JOIN con la tabla pedido

```
SELECT domiciliario.nombre_domiciliario, pedido.id_pedido
FROM domiciliario
INNER JOIN pedido ON pedido.id_domiciliario_pedido = domiciliario.id_domiciliario;
```

nombre_domiciliario	id_pedido
Domiciliario 1	PED001
Domiciliario 2	PED002

Consulta para ver los clientes y las id de los pedidos que han realizado

Para esta consulta se uso un SELECT a cliente.nombre_cliente, y un GROUP_CONCAT a pedido.id_pedido para concatenar todos los ids de los pedidos hechos por un cliente. Además un LEFT JOIN desde la tabla cliente a la tabla pedido

```
SELECT cliente.nombre_cliente, GROUP_CONCAT(pedido.id_pedido SEPARATOR ', ') as pedidos_realizados
FROM cliente
LEFT JOIN pedido ON cliente.cedula = pedido.cedula_cliente
GROUP BY cliente.cedula;
```

Juan Pérez	PED001
María Gómez	PED002

6.0 Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).

Se hizo la vista para ver el nombre de cada cliente con toda su información de contacto, se hizo esta vista porque para Don Pepe puede ser importante tener a la mano esta información de sus clientes.

```
CREATE VIEW contacto_cliente AS
SELECT cliente.nombre_cliente AS nombre, telefono_cliente.numero_cliente AS telefono, cliente.email AS correo, cliente.direccion_cliente AS direccion
FROM cliente
INNER JOIN telefono_cliente ON telefono_cliente.cedula_cliente = cliente.cedula;

SELECT *
FROM contacto_cliente;
```

nombre	telefono	correo	direccion
Juan Pérez	555-1234	juanperez@email.com	Calle 123, Ciudad ABC
María Gómez	555-5678	mariagomez@email.com	Avenida XYZ, Ciudad DEF

Se hizo una vista para ver el listado de productos por su categoría y proveedor puede ser importante para clasificar los productos.

```
CREATE VIEW info_productos AS
SELECT producto.nombre_producto, categoria.nombre_categoria, proveedor.nombre_proveedor
FROM producto
INNER JOIN categoria ON categoria.id_categoria = producto.id_categoria_producto
INNER JOIN proveedor ON proveedor.id_proveedor = producto.id_proveedor_producto;
```

nombre_producto	nombre_categoria	nombre_proveedor
Producto A	Categoría A	Proveedor A
Producto B	Categoría B	Proveedor B

Se hizo una vista para ver el pedido que tiene cada domiciliario con la dirección de entrega, esta vista puede ser importante para Don Pepe para tener un control de sus envíos.

```
196 •   CREATE VIEW info_domicilio AS
197     SELECT domiciliario.nombre_domiciliario, pedido.id_pedido, pedido.direccion_pedido
198     FROM domiciliario
199     INNER JOIN pedido ON pedido.id_domiciliario_pedido = domiciliario.id_domiciliario;
```

nombre_domiciliario	id_pedido	direccion_pedido
Domiciliario 1	PED001	Calle 123, Ciudad ABC
Domiciliario 2	PED002	Avenida XYZ, Ciudad DEF

Se hizo una vista para que Don Pepe pudiera ver los pedidos con los productos y cantidades de productos que llevan. Esto puede ser útil para Don Pepe cuando quiera hacer un inventario.

```
SELECT pedido.id_pedido, GROUP_CONCAT( producto.nombre_producto, ' (', producto_pedido.Cantidad, ') ') as productos_cantidad
FROM producto_pedido
INNER JOIN pedido ON producto_pedido.id_pedido_producto = pedido.id_pedido
INNER JOIN producto ON producto_pedido.id_producto_pedido = producto.id_producto
GROUP BY pedido.id_pedido;
```

id_pedido	productos_cantidad
PED001	Producto A (2) ,Producto B (5)
PED002	Producto B (3)

7.0 Generar al menos 4 procedimientos almacenados.

Procedimiento para agregar un número de teléfono a un cliente

Este procedimiento tiene dos parámetros de entrada " cedula_procedimiento " y " numero_procedimiento ", ambos de tipo VARCHAR. El procedimiento empieza con la declaración BEGIN y finaliza con END y dentro se realiza un INSERT en la tabla teléfono_cliente para las columnas cedula_cliente" y "numero_cliente.

```
217    DELIMITER //
218    CREATE PROCEDURE agregar_telefono_cliente(
219      IN cedula_procedimiento VARCHAR(15),
220      IN numero_procedimiento VARCHAR(15)
221    )
222    BEGIN
223      INSERT INTO telefono_cliente(cedula_cliente, numero_cliente)
224      VALUES(cedula_procedimiento, numero_procedimiento);
225    END//
226    // DELIMITER;
227
228    CALL agregar_telefono_cliente('123456789', '555-5555')
```

	cedula_cliente	numero_cliente
▶	123456789	555-1234
	123456789	555-5555
	987654321	555-5678
*	NULL	NULL

Procedimiento para actualizar un número de teléfono

Este procedimiento tiene tres parámetros de entrada "cedula_procedimiento", "numero_procedimiento" y "nuevo_numero_procedimiento", todos de tipo VARCHAR. El procedimiento empieza con la declaración BEGIN y finaliza con END y dentro se realiza un UPDATE en la tabla teléfono_cliente Y un SET al campo número_cliente para darle el nuevo valor. Luego se usa un WHERE para dar las condiciones para buscar el campo que se le va dar el nuevo valor

```sql
-- procedimiento para actualizar el telefono de un cliente
DELIMITER //
CREATE PROCEDURE actualizar_telefono_cliente (
IN cedula_procedimiento VARCHAR(15),
IN numero_procedimiento VARCHAR(15),
IN nuevo_numero_procedimiento VARCHAR(15)
)
BEGIN
UPDATE telefono_cliente
SET numero_cliente = nuevo_numero_procedimiento
WHERE cedula_cliente = cedula_procedimiento AND numero_cliente = numero_procedimiento;
END //
//DELIMITER ;

CALL actualizar_telefono_cliente ('123456789','555-5555','666-6666')
```

cedula_cliente	numero_cliente
123456789	555-1234
123456789	666-6666
987654321	555-5678
NULL	NULL

Procedimiento para consultar número de teléfonos del cliente

Este procedimiento tiene un parámetros de entrada "cedula_procedimiento", de tipo VARCHAR. El procedimiento empieza con la declaración BEGIN y finaliza con END y dentro se realiza un SELECT numero_cliente Y un FROM a la tabla teléfono_cliente. Luego se usa un WHERE para dar la condiciones para buscar el campo.

```sql
249     -- Procedimiento para consultar numero de telefonos del cliente
250     DELIMITER //
251     CREATE PROCEDURE consultar_telefono_cliente (
252     IN cedula_procedimiento VARCHAR(15)
253     )
254     BEGIN
255     SELECT numero_cliente
256     FROM telefono_cliente
257     WHERE cedula_cliente = cedula_procedimiento;
258     END //
259     //DELIMITER ;
260
261     CALL consultar_telefono_cliente ('123456789')
```

numero_cliente
555-1234
666-6666

Procedimiento para eliminar número de telefono del cliente

Este procedimiento tiene dos parámetros de entrada "cedula_procedimiento", "numero_procedimiento" todos de tipo VARCHAR. El procedimiento empieza con la declaración BEGIN y finaliza con END y dentro se realiza un DELETE desde teléfono_cliente Y Luego se usa un WHERE para dar las condiciones para buscar el campo que se desea eliminar

```
265  CREATE PROCEDURE eliminar_telefono_cliente (
266  IN cedula_procedimiento VARCHAR(10),
267  IN numero_procedimiento VARCHAR(15)
268  )
269  BEGIN
270  DELETE FROM telefono_cliente
271  WHERE cedula_cliente = cedula_procedimiento AND numero_cliente = numero_procedimiento;
272  END //
273  //DELIMITER ;
274
275  CALL eliminar_telefono_cliente ('123456789','666-6666')
```

	cedula_cliente	numero_cliente
▶	123456789	555-1234
	987654321	555-5678
*	NULL	NULL

8.0 Generar al menos 4 triggers

Trigger para disminuir el stock cuando se realice una compra en la tabla producto_pedido.

Este trigger se dispra cuando insertan información en la tabla producto_pedido, donde hace una validación, si la cantidad es mayor a el stock del producto entonces envia un mensaje de error que dice que 'No hay suficiente stock para realizar el pedido', en caso de que la cantidad sea menor al stock entonces hace un UPDATE al stock restándole la cantidad.

```
279 •  CREATE TRIGGER disminuir_stock AFTER INSERT ON producto_pedido
280    FOR EACH ROW
281    BEGIN
282      IF NEW.Cantidad > (SELECT stock FROM producto WHERE id_producto = NEW.id_producto_pedido) THEN
283        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'No hay suficiente stock para realizar el pedido';
284      ELSE
285        UPDATE producto SET stock = stock - NEW.Cantidad WHERE id_producto = NEW.id_producto_pedido;
286      END IF;
287    END;
288    // DELIMITER ;
289
290 •  INSERT INTO producto_pedido(id_producto_pedido, id_pedido_producto, Cantidad)
291    VALUES('prod001', 'PED001', 200)
292
```

trigger para validar que el domiciliario este en la misma zona que el cliente

Este trigger se dispara cuando insertan información en la tabla pedido, donde se valida con un IF si la zona del cliente es diferente que la zona del domiciliario, si es así, el trigger dispara un error que dice que el domiciliario no esta en la misma zona que el cliente.

```
295 •  CREATE TRIGGER validar_zona_domiciliario
296    BEFORE INSERT ON pedido
297    FOR EACH ROW
298    BEGIN
299        DECLARE trigger_zona_cliente VARCHAR(15);
300        DECLARE trigger_zona_domiciliario VARCHAR(15);
301
302        SELECT zona_cliente INTO trigger_zona_cliente FROM cliente WHERE cedula = NEW.cedula_cliente;
303        SELECT zona_domiciliario INTO trigger_zona_domiciliario FROM domiciliario WHERE id_domiciliario = NEW.id_domiciliario_pedido;
304
305        IF trigger_zona_cliente != trigger_zona_domiciliario THEN
306            SIGNAL SQLSTATE '45000'
307            SET MESSAGE_TEXT = 'El domiciliario no está en la misma zona que el cliente';
308        END IF;
309    END;
310    // DELIMITER ;
```

Se crea un trigger con el nombre agregar_teléfono que se dispara cuando inserten información en la tabla teléfono_cliente y dejara el registro en la tabla control_de_cambios_tienda

```sql
-- Tabla control_de_cambios_librería
CREATE TABLE IF NOT EXISTS control_de_cambios_tienda (
    usuario VARCHAR(50),
    accion VARCHAR(50),
    fecha DATETIME DEFAULT current_timestamp
);

-- Trigger agregar
DELIMITER //
CREATE TRIGGER agregar_telefono AFTER INSERT ON telefono_cliente
    FOR EACH ROW
    BEGIN
        INSERT INTO control_de_cambios_tienda VALUES(user(), 'agregar', now());
    END;
// DELIMITER ;
```

Se crea un trigger con el nombre eliminar_teléfono que se dispara cuando elimine información en la tabla teléfono_cliente y dejara el registro en la tabla control_de_cambios_tienda

```sql
-- Trigger eliminar
DELIMITER //
CREATE TRIGGER eliminar_telefono AFTER DELETE ON telefono_cliente
    FOR EACH ROW
    BEGIN
        INSERT INTO control_de_cambios_tienda VALUES(user(), 'eliminar', now());
    END;
// DELIMITER ;
```

usuario	accion	fecha
root@localhost	agregar	2023-02-17 10:05:11
root@localhost	eliminar	2023-02-17 10:05:32

9.0 Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.

Creamos una clase llamada MySqlConnector.

La clase contiene los siguientes atributos

```
public class MySqlConnector {  Complexity is 9 It'
    private String connectionString;
    private String dbName = "";
    private String dbUser = "";
    private String dbPassword = "";
```

Estos atributos sirven para setear los datos de la base de datos, como lo son el nombre que tiene la base de datos que en este caso es "tienda" y el usuario y la contraseña mysql.

Se crea el método constructor:

```
public MySqlConnector (String dbName, String dbUser, String dbPassword){
    this.dbName = dbName;
    this.dbUser = dbUser;
    this.dbPassword = dbPassword;
}
```

Se crea el método connect que sirve para abrir la conexión con la base de datos

```
public void connect(){  Complexity is 8 It's time to do something...
    try{
        if(this.connection == null){
            Class.forName( className: "com.mysql.cj.jdbc.Driver");
            System.out.println("Conectándose a mysql");
            this.connectionString = "jdbc:mysql://localhost:3306/" + this.dbName;
            this.connection = DriverManager.getConnection(
                    this.connectionString,
                    this.dbUser,
                    this.dbPassword
            );
            if (this.connection == null){
                System.out.println("Error de conexión");
            } else {
                System.out.println("Conexión exitosa");
            }
        }
    }catch(ClassNotFoundException e){
        e.printStackTrace();
    }catch(SQLException e){
        e.printStackTrace();
    }
}
```

Se crea el método closeConnection el cual sirve para cerrar la conexión con la base de datos

```
public void closeConnection (){  Complexity is 4 Everything is cool!
    try {
        if(this.connection != null){
            this.connection.close();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

Se crea el método getStatement el cual se usa para ejecutar querys (SELECT, INSERT, UPDATE, DELETE).

```
MiguelMendoza22
public PreparedStatement getStatement(String sql){  Complexity is 4 Everything is cool!
    try{
        return this.connection.prepareStatement(sql);
    }catch (SQLException e){
        e.printStackTrace();
        return null;
    }
}
```

Se crea la clase tienda. Esta clase va contener todos los métodos para poblar todas las tablas de la base de datos.

Se crea un objeto mySqlConnector a partir de la clase MySqlConnector y se crea el constructor para inicializar el objeto.

```
public class Tienda {  Complexity is 11 You must be kidding
    MySqlConnector mySqlConnector;

    MiguelMendoza22
    public Tienda(MySqlConnector mySqlConnector) {
        this.mySqlConnector = mySqlConnector;
    }
```

Se crea el método para insertar clientes, se usa la librería Faker para generar datos aleatorios para los campos de la tabla cliente

```
public void insertarCliente() {  Complexity is 7 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO cliente(cedula, nombre_cliente, direccion_cliente, zona_cliente, email, password_cliente)" +
                "VALUES (?, ?, ?, ?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        int cc = 1999999;
        for(int i = 0; i < 50; i++){
            statement.setString( parameterIndex: 1,  x: (cc + i) + "");
            statement.setString( parameterIndex: 2, faker.name().firstName());
            statement.setString( parameterIndex: 3, faker.address().streetAddressNumber());
            statement.setString( parameterIndex: 4, faker.options().option( ...options: "Zona 1", "Zona 2", "Zona 3", "Zona 4"));;
            statement.setString( parameterIndex: 5, faker.internet().emailAddress());
            statement.setString( parameterIndex: 6,faker.internet().password());
            statement.execute();
        }
        statement.close();
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

cedula	nombre_cliente	direccion_cliente	zona_cliente	email	password_cliente
1999999	Stephan	683	Zona 4	reid.mitchell@gmail.com	zhvr8579b3k6e2
2000000	Lorilee	899	Zona 3	shari.witting@yahoo.com	wbqvdju4yn3s7d
2000001	Signe	725	Zona 2	paris.hammes@yahoo.com	bhe1w2lxxe4
2000002	Jona	171	Zona 1	detra.reichel@gmail.com	698yf3zsp
2000003	Madonna	57	Zona 3	lue.rosenbaum@hotmail.com	ess57xfyje1g
2000004	Diedra	495	Zona 3	kalyn.oconner@yahoo.com	3kspbfeadk3mft
2000005	Princess	355	Zona 1	santana.stark@hotmail.com	d52fex6f
2000006	Tonisha	724	Zona 3	trina.kutch@hotmail.com	76kihbhbh07
2000007	Benjamin	402	Zona 3	cheree.fay@hotmail.com	4dz3n2vctrc
2000008	Parthenia	118	Zona 1	jimmie.von@yahoo.com	kbvfyus3gqd9
2000009	Madalyn	163	Zona 1	lessie.willms@hotmail.com	9j4ftdgklu3z
2000010	Desirae	428	Zona 4	versie.sauer@yahoo.com	1rz9yi24i
2000011	Phoebe	914	Zona 2	yong.cartwright@yahoo.com	ktgotkfih2l
2000012	Pamala	629	Zona 1	jarrett.ruecker@yahoo.com	ubbn80uajw2b3m
2000013	Lyn	870	Zona 2	mila.schumm@hotmail.com	0il333dnwcwul
2000014	George	486	Zona 2	collin.daugherty@hotmail.com	5wre22btcg73p
2000015	Melodee	678	Zona 2	cami.williamson@hotmail.com	i9ag7avb4
2000016	Charmaine	634	Zona 1	aleida.mcglynn@hotmail.com	f4gc5xkrreihv
2000017	Burton	405	Zona 3	liliana.heaney@yahoo.com	nh07eqhy
2000018	Jeremiah	86	Zona 2	jose.mohr@yahoo.com	cnsnw2eqzugc08k
2000019	Lurline	849	Zona 4	arianne.funk@gmail.com	7d1f62vbd7wwx4

Se crea el método para insertar datos a la tabla teléfono cliente, se usa la librería faker para generar números de teléfonos aleatorios.

```java
public void insertarTelefonoCliente() {  Complexity is 7 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO telefono_cliente (cedula_cliente, numero_cliente)" +
                "VALUES (?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        int cc = 1999999;
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: (cc + i) + "");
            statement.setString( parameterIndex: 2,faker.phoneNumber().cellPhone());
            statement.execute();
        }
        statement.close();
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

cedula_cliente	numero_cliente
1999999	(645) 375-8015
2000000	689-648-4195
2000001	963-996-6278
2000002	(852) 889-1045
2000003	1-415-243-3627
2000004	1-992-051-2393
2000005	(233) 586-0223
2000006	(421) 385-4234
2000007	1-231-100-8648
2000008	1-035-983-1297
2000009	(723) 153-7143
2000010	1-766-735-6442
2000011	385.809.1403
2000012	1-578-622-2120
2000013	1-769-132-4802
2000014	1-459-410-9196

Se crea el método para insertar datos en la tabla proovedor, se usa la librería faker para generar nombres aleatorios de proveedores.

```java
public void insertarProveedor() {  Complexity is 6 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO proveedor (id_proveedor, nombre_proveedor)" +
                "VALUES (?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1, x: "prov"+i);
            statement.setString( parameterIndex: 2,faker.company().name());
            statement.execute();
        }
        statement.close();
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

id_proveedor	nombre_proveedor
prov0	Ziemann, Flatley and Thiel
prov1	Considine-Casper
prov10	Mills, Ferry and Mayert
prov11	Tillman Inc
prov12	Sawayn-Leuschke
prov13	Smith-Kohler
prov14	Botsford LLC
prov15	Fay Group
prov16	Blick and Sons
prov17	Ebert, Kohler and Hane
prov18	Nitzsche-Moore
prov19	Stoltenberg and Sons
prov2	O'Kon and Sons
prov20	Conroy, Wolf and Hirthe
prov21	Smith Group
prov22	Kris, Cormier and Emard
prov23	Gottlieb, Auer and Lowe

Se crea el método para insertar datos a la tabla teléfono proveedor, se usa la librería faker para generar números de teléfonos aleatorios.

```java
public void insertarTelefonoProveedor() {  Complexity is 6 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO telefono_proveedor (id_proveedor_telefono, numero_proveedor)" +
                "VALUES (?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1, x: "prov"+i);
            statement.setString( parameterIndex: 2,faker.phoneNumber().cellPhone());
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

	id_proveedor_telefono	numero_proveedor
▶	prov0	051-352-3307
	prov1	928-761-1930
	prov10	085-429-1145
	prov11	1-193-377-9516
	prov12	(069) 122-7125
	prov13	433.689.3181
	prov14	1-809-392-4268
	prov15	468-241-6598
	prov16	068.001.2499
	prov17	507.024.1889
	prov18	(789) 553-0251
	prov19	129.741.7722
	prov2	094.573.0904
	prov20	(227) 038-4211
	prov21	786.989.5995
	prov22	1-970-601-8678
	prov23	478-022-8970
	prov24	(742) 112-1264

Se crea el método para insertar datos a la tabla categoría.

```java
public void insertarCategoria() {  Complexity is 7 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO categoria (id_categoria, nombre_categoria, condiciones_almacenaimiento, observaciones)" +
                "VALUES (?, ?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: "cat" + i);
            statement.setString( parameterIndex: 2, faker.commerce().department());
            statement.setString( parameterIndex: 3, faker.options().option( ...options: "Frio", "Seco", "Humedo", "Caliente"));
            statement.setString( parameterIndex: 4,  x: "observaciones" + i);
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

id_categoria	nombre_categoria	condiciones_almacenaimiento	observaciones
cat0	Books & Outdoors	Caliente	observaciones0
cat1	Movies	Caliente	observaciones1
cat10	Kids	Humedo	observaciones10
cat11	Computers	Caliente	observaciones11
cat12	Automotive, Clothing & Movies	Seco	observaciones12
cat13	Games	Humedo	observaciones13
cat14	Outdoors & Sports	Seco	observaciones14
cat15	Garden & Industrial	Humedo	observaciones15
cat16	Garden, Grocery & Tools	Caliente	observaciones16
cat17	Sports	Caliente	observaciones17
cat18	Computers	Seco	observaciones18
cat19	Baby & Health	Frio	observaciones19
cat2	Baby & Computers	Humedo	observaciones2
cat20	Computers	Humedo	observaciones20
cat21	Home & Toys	Humedo	observaciones21
cat22	Health	Seco	observaciones22

Se crea el método para insertar datos a la tabla producto

```java
public void insertarProducto() {  Complexity is 8 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO producto (id_producto, nombre_producto, marca, origen, contenido, fotografia, stock, " +
                "id_categoria_producto, id_proveedor_producto)" +  "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: "prod"+i);
            statement.setString( parameterIndex: 2, faker.commerce().productName());
            statement.setString( parameterIndex: 3, faker.company().name());
            statement.setString( parameterIndex: 4, faker.country().name());
            statement.setString( parameterIndex: 5, faker.options().option( ...options: "1 kg","500 gr", "350 ml", "1 lt"));
            statement.setString( parameterIndex: 6,  x: "fotografia"+i);
            statement.setString( parameterIndex: 7, String.valueOf(faker.number().numberBetween(20, 60)));
            statement.setString( parameterIndex: 8,  x: "cat" + i);
            statement.setString( parameterIndex: 9, x: "prov"+i);
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

id_producto	nombre_producto	marca	origen	contenido	fotografia	stock	id_categoria_producto	id_proveedor_producto
prod0	Practical Bronze Table	Lowe, Wilderman and Zboncak	United Arab Emirates	350 ml	fotografia0	37	cat0	prov0
prod1	Fantastic Marble Keyboard	Dibbert-Friesen	Samoa	500 gr	fotografia1	43	cat1	prov1
prod10	Rustic Wooden Coat	Swift, Purdy and Schneider	Denmark	350 ml	fotografia10	41	cat10	prov10
prod11	Enormous Copper Keyboard	Leuschke Group	Belarus	1 lt	fotografia11	52	cat11	prov11
prod12	Enormous Rubber Lamp	Lynch Inc	Belize	1 lt	fotografia12	40	cat12	prov12
prod13	Intelligent Paper Lamp	Witting and Sons	Syrian Arab Republic	500 gr	fotografia13	57	cat13	prov13
prod14	Ergonomic Steel Keyboard	Hodkiewicz-Daugherty	Thailand	1 kg	fotografia14	46	cat14	prov14
prod15	Small Steel Bench	Nienow, Lynch and Gleason	Haiti	500 gr	fotografia15	56	cat15	prov15
prod16	Sleek Rubber Knife	Hoeger-White	Korea (Democratic People's Republic of)	350 ml	fotografia16	31	cat16	prov16
prod17	Fantastic Marble Shirt	Wintheiser Group	Marshall Islands	500 gr	fotografia17	45	cat17	prov17
prod18	Gorgeous Rubber Knife	McCullough, Morar and Heaney	Mali	1 lt	fotografia18	27	cat18	prov18
prod19	Ergonomic Plastic Gloves	Goyette and Sons	Guinea-Bissau	350 ml	fotografia19	53	cat19	prov19
prod2	Small Linen Gloves	Jast-Upton	Belarus	500 gr	fotografia2	29	cat2	prov2
prod20	Incredible Linen Gloves	Armstrong and Sons	Germany	350 ml	fotografia20	26	cat20	prov20
prod21	Synergistic Aluminum Plate	Spencer, Renner and Hagenes	Angola	350 ml	fotografia21	55	cat21	prov21
prod22	Mediocre Copper Wallet	Pouros and Sons	Panama	1 kg	fotografia22	30	cat22	prov22
prod23	Synergistic Linen Bottle	Bernhard-Waters	Serbia	500 gr	fotografia23	48	cat23	prov23
prod24	Small Wool Hat	Greenfelder and Sons	Hungary	1 kg	fotografia24	23	cat24	prov24
prod25	Ergonomic Aluminum Watch	Fahey, Wunsch and Pfannerstill	Singapore	1 kg	fotografia25	32	cat25	prov25

Método para insertar domiciliarios

```java
public void insertarDomiciliario() {  Complexity is 6 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO domiciliario (id_domiciliario, nombre_domiciliario, numero_matricula, zona_domiciliario)" +
                "VALUES (?, ?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: "dom"+i);
            statement.setString( parameterIndex: 2,faker.name().firstName());
            statement.setString( parameterIndex: 3, faker.regexify("[A-Z0-9]{6}"));
            statement.setString( parameterIndex: 4, faker.options().option( ...options: "Zona 1", "Zona 2", "Zona 3", "Zona 4"));
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

	id_domiciliario	nombre_domiciliario	numero_matricula	zona_domiciliario
▶	dom0	Margarete	K1XR99	Zona 1
	dom1	Charley	OGA04D	Zona 3
	dom10	Su	M3OQNY	Zona 4
	dom11	Ned	94T122	Zona 2
	dom12	Nadine	79SUCO	Zona 2
	dom13	Cyndi	13Q333	Zona 3
	dom14	Jarvis	2KPN3N	Zona 1
	dom15	Hilton	W8QLGJ	Zona 2
	dom16	Rhea	63B314	Zona 2
	dom17	Herschel	0J420Q	Zona 3
	dom18	Lynne	FN484V	Zona 3
	dom19	Matha	526Y7R	Zona 3
	dom2	Carmine	09010X	Zona 3
	dom20	Myung	LMW0LT	Zona 1
	dom21	Reynaldo	OAH8L2	Zona 3
	dom22	Rodolfo	MY1PP6	Zona 3
	dom23	Victor	2NGT80	Zona 3
	dom24	Angel	6P8U5H	Zona 4
	dom25	Will	0197G9	Zona 4

Método para insertar pedidos

```
public void insertarPedido() {  Complexity is 9 It's time to do something...
    try {
        Faker faker = new Faker();
        LocalDate fechaActual = LocalDate.now();
        String query = "INSERT INTO pedidos (id_pedido, direccion_pedido, fecha_pedido, numero_tarjeta, fecha_caducidad_tarjeta, " +
                "total_valor, cedula_cliente, id_domiciliario_pedido)" +
                "VALUES (?, ?, ?, ?, ?, ?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        int cc = 1999999;
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: "ped" + i);
            statement.setString( parameterIndex: 2, faker.address().streetAddressNumber());
            statement.setString( parameterIndex: 3, String.valueOf(fechaActual));
            statement.setString( parameterIndex: 4,  x: "1000000" + i);
            statement.setString( parameterIndex: 5,  x: "05/22");
            statement.setString( parameterIndex: 6,String.valueOf(faker.number().numberBetween(10000, 100000)));
            statement.setString( parameterIndex: 7,  x: (cc + i) + "");
            statement.setString( parameterIndex: 8,  x: "dom"+i);
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

id_pedido	direccion_pedido	fecha_pedido	numero_tarjeta	fecha_caducidad_tarjeta	total_valor	cedula_cliente	id_domiciliario_pedido
ped0	991	2023-02-19 00:00:00	10000000	05/22	72655	1999999	dom0
ped1	112	2023-02-19 00:00:00	10000001	05/22	21359	2000000	dom1
ped10	408	2023-02-19 00:00:00	100000010	05/22	25368	2000009	dom10
ped11	673	2023-02-19 00:00:00	100000011	05/22	89514	2000010	dom11
ped12	75	2023-02-19 00:00:00	100000012	05/22	66339	2000011	dom12
ped13	317	2023-02-19 00:00:00	100000013	05/22	30184	2000012	dom13
ped14	47	2023-02-19 00:00:00	100000014	05/22	73368	2000013	dom14
ped15	593	2023-02-19 00:00:00	100000015	05/22	61063	2000014	dom15
ped16	53	2023-02-19 00:00:00	100000016	05/22	73555	2000015	dom16
ped17	924	2023-02-19 00:00:00	100000017	05/22	57283	2000016	dom17
ped18	89	2023-02-19 00:00:00	100000018	05/22	28242	2000017	dom18
ped19	572	2023-02-19 00:00:00	100000019	05/22	58489	2000018	dom19
ped2	956	2023-02-19 00:00:00	10000002	05/22	61928	2000001	dom2
ped20	25	2023-02-19 00:00:00	100000020	05/22	99041	2000019	dom20
ped21	185	2023-02-19 00:00:00	100000021	05/22	12552	2000020	dom21
ped22	95	2023-02-19 00:00:00	100000022	05/22	29381	2000021	dom22
ped23	341	2023-02-19 00:00:00	100000023	05/22	90341	2000022	dom23
ped24	235	2023-02-19 00:00:00	100000024	05/22	40543	2000023	dom24
ped25	991	2023-02-19 00:00:00	100000025	05/22	98962	2000024	dom25

Método para insertar datos a la tabla producto_pedido

```
public void insertarProductoPedido() {  Complexity is 7 It's time to do something...
    try {
        Faker faker = new Faker();
        String query = "INSERT INTO producto_pedido(id_producto_pedido, id_pedido_producto, Cantidad)" +
                "VALUES (?, ?, ?);";
        PreparedStatement statement = this.mySqlConnector.getStatement(query);
        for (int i = 0; i < 50; i++) {
            statement.setString( parameterIndex: 1,  x: "prod"+i);
            statement.setString( parameterIndex: 2,  x: "ped" + i);
            statement.setString( parameterIndex: 3, String.valueOf(faker.number().numberBetween(1, 15)));
            statement.execute();
        }
    }catch (SQLException e){
        e.printStackTrace();
    }
}
```

id_producto_pedido	id_pedido_producto	Cantidad
prod0	ped0	7
prod1	ped1	7
prod2	ped2	5
prod3	ped3	8
prod4	ped4	11
prod5	ped5	10
prod6	ped6	10
prod7	ped7	11
prod8	ped8	5
prod9	ped9	3
prod10	ped10	6
prod11	ped11	5
prod12	ped12	7
prod13	ped13	3
prod14	ped14	10
prod15	ped15	2

10.0 Al terminar el ejercicio responda ¿Está conforme con el resultado obtenido según el contexto o cree que hubiera obtenido un mejor resultado con una base de datos no relacional?

Siendo sincero, nunca he usado una base de datos no relacional, pero investigando un poco creo que el mejor resultado para este contexto se consigue con base de datos relacionales, ya que los datos para el ejercicio de la tienda son estructurados, además que son perfectas para contabilidad, inventarios y facturaciones, que son algunas de las necesidades que tiene Don Pepe, además que son perfectas para relaciones entre tablas como se necesitaba en el ejercicio.

En conclusión, una base de datos relacional podría ser la mejor opción si necesitas garantizar la integridad de los datos y tener la capacidad de realizar consultas complejas. Los sistemas de bases de datos relacionales están diseñados para manejar grandes cantidades de datos estructurados y pueden ser muy útiles en aplicaciones que requieren transacciones, como el control de inventario y ventas.