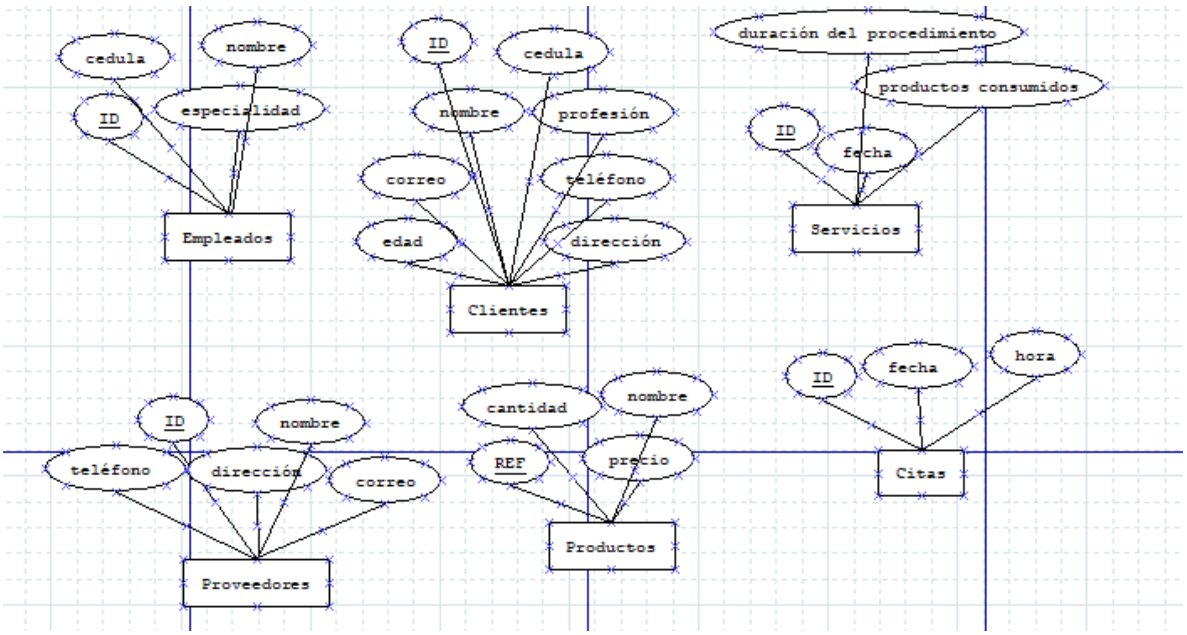
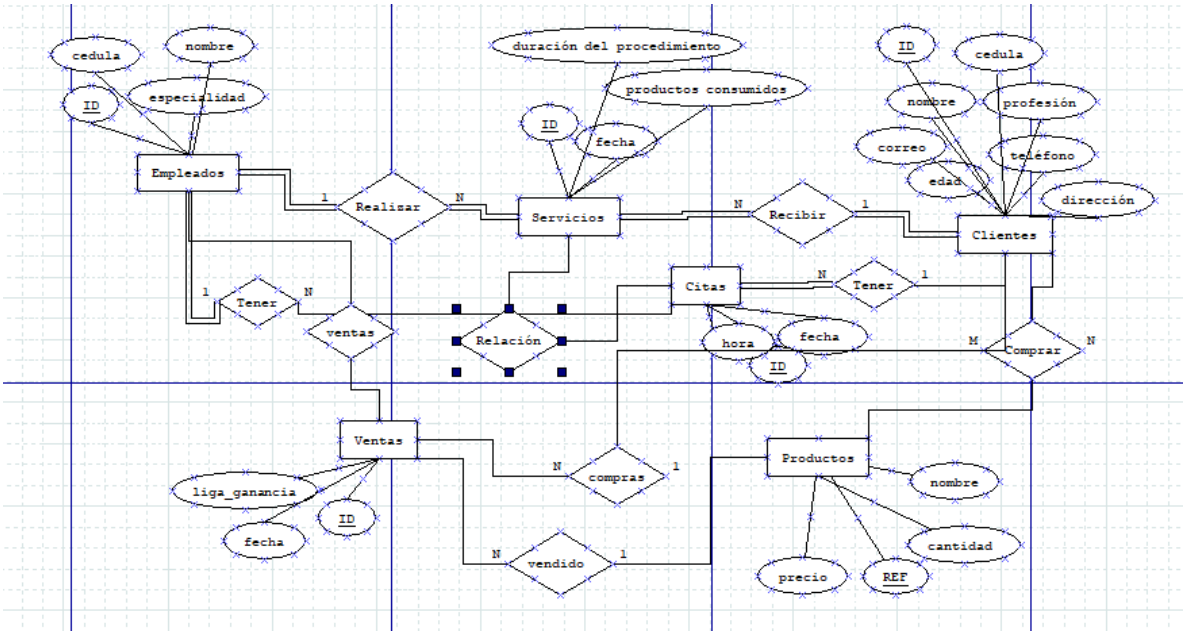
Barbería (Ejercicio A)

1. Generando tablas y atributos diagrama E-R



Generando tablas y atributos diagrama E-R y relaciones



Las relaciones entre las entidades son las siguientes:

- La entidad "Empleados" tiene una relación de uno a muchos con la entidad "Servicios", ya que un empleado puede prestar muchos servicios, pero un servicio solo puede ser prestado por un empleado.

- La entidad "Clientes" tiene una relación de uno a muchos con la entidad "Citas", ya que un cliente puede tener muchas citas, pero una cita solo puede ser programada para un cliente.

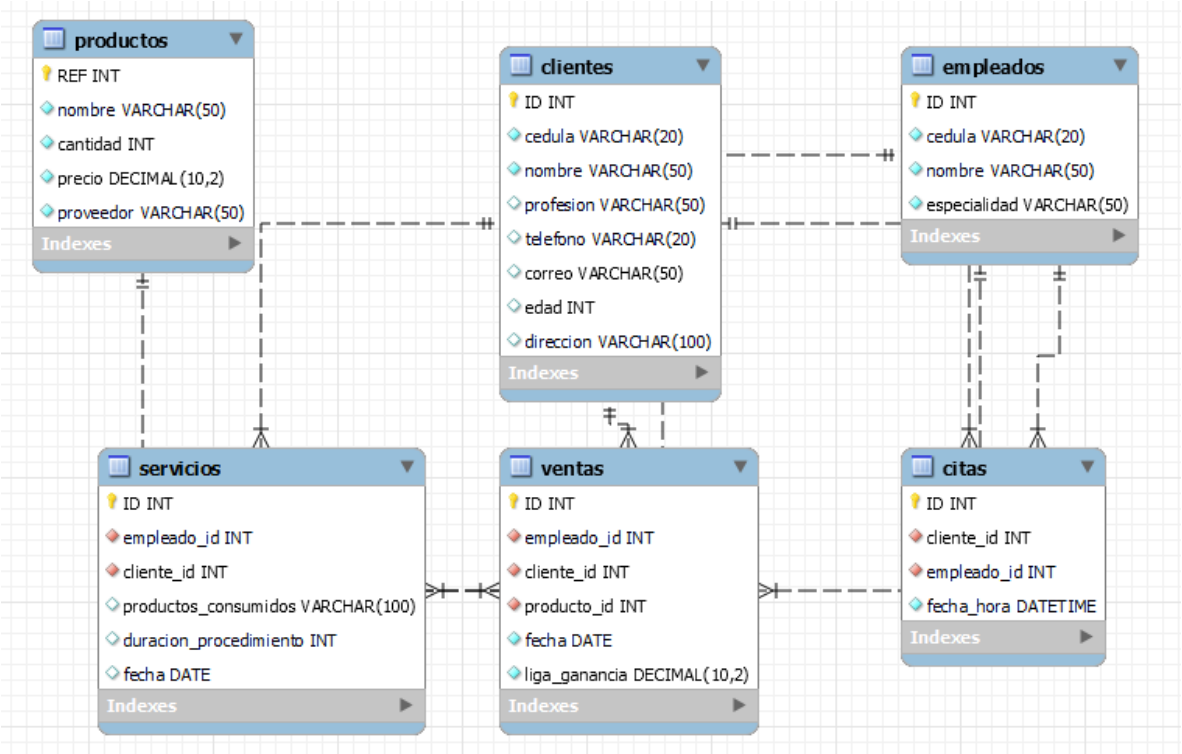
- La entidad "Empleados" tiene una relación de uno a muchos con la entidad "Citas", ya que un empleado puede tener muchas citas, pero una cita solo puede ser asignada a un empleado.

- La entidad "Productos" tiene una relación de uno a muchos con la entidad "Ventas", ya que un producto puede ser vendido muchas veces, pero una venta solo puede incluir un producto.

- La entidad "Empleados" tiene una relación de uno a muchos con la entidad "Ventas", ya que un empleado puede hacer muchas ventas, pero una venta solo puede ser realizada por un empleado.

- La entidad "Clientes" tiene una relación de uno a muchos con la entidad "Ventas", ya que un cliente puede hacer muchas compras, pero una compra solo puede ser realizada por un cliente.

Modelo relación:

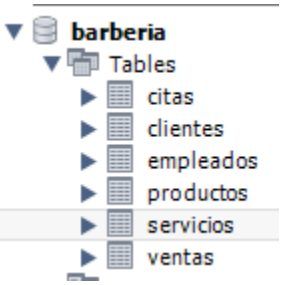


Consulta para crear la base datos :

```
CREATE DATABASE barberia;

USE barberia;

CREATE TABLE empleados (
    ID INT NOT NULL AUTO_INCREMENT,
    cedula VARCHAR(20) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    especialidad VARCHAR(50) NOT NULL,
    PRIMARY KEY (ID)
);

CREATE TABLE clientes (
    ID INT NOT NULL AUTO_INCREMENT,
    cedula VARCHAR(20) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    profesion VARCHAR(50),
    telefono VARCHAR(20),
    correo VARCHAR(50),
    edad INT,
    direccion VARCHAR(100),
    PRIMARY KEY (ID)
);
```

```
CREATE TABLE servicios (
    ID INT NOT NULL AUTO_INCREMENT,
    empleado_id INT NOT NULL,
    cliente_id INT NOT NULL,
    productos_consumidos VARCHAR(100),
    duracion_procedimiento INT,
    fecha DATE,
    PRIMARY KEY (ID),
    FOREIGN KEY (empleado_id) REFERENCES empleados(ID),
    FOREIGN KEY (cliente_id) REFERENCES clientes(ID)
);

CREATE TABLE citas (
    ID INT NOT NULL AUTO_INCREMENT,
    cliente_id INT NOT NULL,
    empleado_id INT NOT NULL,
    fecha_hora DATETIME NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (cliente_id) REFERENCES clientes(ID),
    FOREIGN KEY (empleado_id) REFERENCES empleados(ID)
);
```

```sql
CREATE TABLE productos (
    REF INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    proveedor VARCHAR(50) NOT NULL,
    PRIMARY KEY (REF)
);

CREATE TABLE ventas (
    ID INT NOT NULL AUTO_INCREMENT,
    empleado_id INT NOT NULL,
    cliente_id INT NOT NULL,
    producto_id INT NOT NULL,
    fecha DATE NOT NULL,
    liga_ganancia DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (ID),
    FOREIGN KEY (empleado_id) REFERENCES empleados(ID),
    FOREIGN KEY (cliente_id) REFERENCES clientes(ID),
    FOREIGN KEY (producto_id) REFERENCES productos(REF)
);
```



Normalizando base de datos

1FN:

- Para aplicar la primera forma normal a esta base de datos, se debe revisar cada tabla y asegurarse de que cada columna tenga un valor único y que no haya valores repetidos. En caso de que una tabla contenga valores repetidos, estos se deben eliminar y crear una nueva tabla para almacenarlos.

- En este caso, se puede ver que las tablas empleadas, clientes, productos y ventas cumplen con la primera forma normal, ya que cada columna tiene un valor único. Por lo tanto, no es necesario realizar ninguna modificación en estas tablas.

- Sin embargo, la tabla servicios y citas contienen columnas repetidas. La tabla servicios tiene las columnas empleado_id y cliente_id, que se refieren a las mismas entidades que las columnas ID de las tablas empleados y clientes, respectivamente. De manera similar, la tabla citas tiene las columnas cliente_id y empleado_id, que se refieren a las mismas entidades que las columnas ID de las tablas clientes y empleados, respectivamente.

- Para cumplir con la primera forma normal, se debe crear una nueva tabla que contenga las columnas ID de las tablas empleados y clientes, y luego referenciar estas tablas desde las tablas servicios y citas.

- **A continuación se muestra el código para crear las nuevas tablas:**

```sql
CREATE TABLE empleados_clientes (
ID INT NOT NULL AUTO_INCREMENT,
empleado_id INT NOT NULL,
cliente_id INT NOT NULL,
PRIMARY KEY (ID),
FOREIGN KEY (empleado_id) REFERENCES empleados(ID),
FOREIGN KEY (cliente_id) REFERENCES clientes(ID)
);
```

```sql
CREATE TABLE servicios (
ID INT NOT NULL AUTO_INCREMENT,
empleado_cliente_id INT NOT NULL,
productos_consumidos VARCHAR(100),
duracion_procedimiento INT,
fecha DATE,
PRIMARY KEY (ID),
FOREIGN KEY (empleado_cliente_id)
REFERENCES empleados_clientes(ID)
);
```

```sql
CREATE TABLE citas (
ID INT NOT NULL AUTO_INCREMENT,
empleado_cliente_id INT NOT NULL,
fecha_hora DATETIME NOT NULL,
PRIMARY KEY (ID),
FOREIGN KEY (empleado_cliente_id)
REFERENCES empleados_clientes(ID)
);
```

- En la tabla empleados_clientes, se han eliminado las columnas cedula, nombre y especialidad, ya que estas columnas se encuentran en las tablas empleados y clientes. En su lugar, se ha agregado una nueva columna ID para identificar la relación entre empleados y clientes.

- En las tablas servicios y citas, se ha agregado una nueva columna empleado_cliente_id para referenciar a la tabla empleados_clientes. Además, se han eliminado las columnas empleado_id y cliente_id.

- De esta manera, se ha aplicado la primera forma normal a la base de datos barberia.

2FN:

- Para aplicar la Segunda Forma Normal (2FN) a la base de datos "barberia", primero debemos verificar si existe alguna tabla con dependencias parciales en sus columnas. En este caso, la única tabla que puede presentar este problema es la tabla "servicios", ya que la columna "productos_consumidos" depende funcionalmente del identificador del servicio y no del identificador de producto. Esto puede causar redundancia de datos si un mismo producto se consume en múltiples servicios.

- Por lo tanto, podemos aplicar la 2FN descomponiendo la tabla "servicios" en dos tablas: "servicios" y "consumo_productos". La tabla "servicios" contendrá solo la información relacionada con el servicio, mientras que la tabla "consumo_productos" contendrá la información de los productos consumidos en cada servicio.

    - **A continuación, mostramos el código SQL para crear las nuevas tablas:**

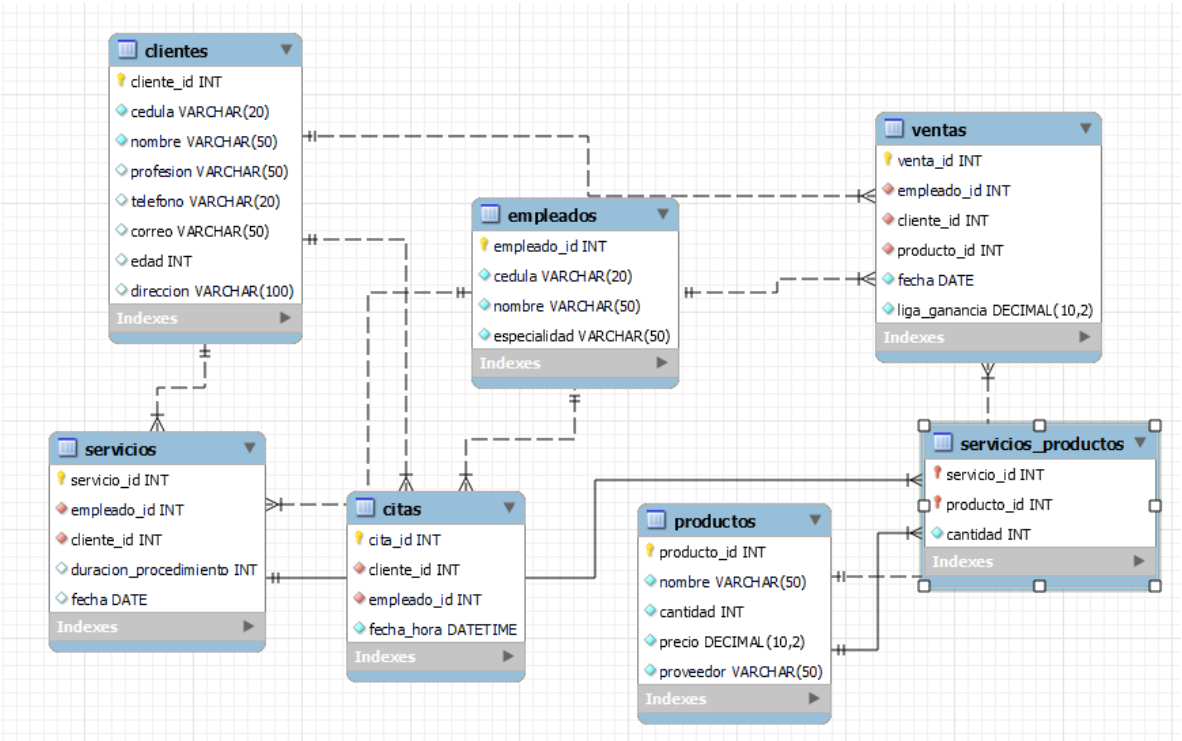
```sql
CREATE TABLE servicios (
    ID INT NOT NULL AUTO_INCREMENT,
    empleado_id INT NOT NULL,
    cliente_id INT NOT NULL,
    duracion_procedimiento INT,
    fecha DATE,
    PRIMARY KEY (ID),
    FOREIGN KEY (empleado_id) REFERENCES empleados(ID),
    FOREIGN KEY (cliente_id) REFERENCES clientes(ID)
);
```

```sql
CREATE TABLE consumo_productos (
    servicio_id INT NOT NULL,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    PRIMARY KEY (servicio_id, producto_id),
    FOREIGN KEY (servicio_id)
    REFERENCES servicios(ID),
    FOREIGN KEY (producto_id)
    REFERENCES productos(REF)
);
```

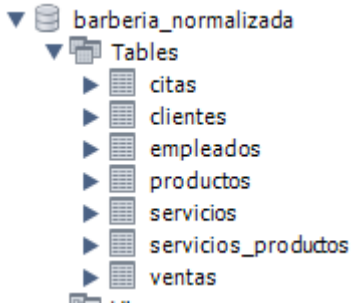
3FN:

Después de aplicar la segunda forma normal, en este caso no es necesario aplicar la tercera forma normal ya que todas las tablas cumplen con las condiciones de esta forma normal. Las tablas tienen claves primarias únicas y no hay dependencias transitivas.

Diagrama de Barberia normalizada:



Consultas para crear la base de datos normalizada



```sql
CREATE DATABASE barberia_normalizada;

USE barberia_normalizada;

CREATE TABLE empleados (
    empleado_id INT NOT NULL AUTO_INCREMENT,
    cedula VARCHAR(20) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    especialidad VARCHAR(50) NOT NULL,
    PRIMARY KEY (empleado_id)
);

CREATE TABLE clientes (
    cliente_id INT NOT NULL AUTO_INCREMENT,
    cedula VARCHAR(20) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    profesion VARCHAR(50),
    telefono VARCHAR(20),
    correo VARCHAR(50),
    edad INT,
    direccion VARCHAR(100),
    PRIMARY KEY (cliente_id)
);
```

```sql
CREATE TABLE servicios (
    servicio_id INT NOT NULL AUTO_INCREMENT,
    empleado_id INT NOT NULL,
    cliente_id INT NOT NULL,
    duracion_procedimiento INT,
    fecha DATE,
    PRIMARY KEY (servicio_id),
    FOREIGN KEY (empleado_id)
    REFERENCES empleados(empleado_id),
    FOREIGN KEY (cliente_id)
    REFERENCES clientes(cliente_id)
);

CREATE TABLE productos (
    producto_id INT NOT NULL AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    proveedor VARCHAR(50) NOT NULL,
    PRIMARY KEY (producto_id)
);
```

```sql
CREATE TABLE ventas (
    venta_id INT NOT NULL AUTO_INCREMENT,
    empleado_id INT NOT NULL,
    cliente_id INT NOT NULL,
    producto_id INT NOT NULL,
    fecha DATE NOT NULL,
    liga_ganancia DECIMAL(10,2) NOT NULL,
    PRIMARY KEY (venta_id),
    FOREIGN KEY (empleado_id)
    REFERENCES empleados(empleado_id),
    FOREIGN KEY (cliente_id)
    REFERENCES clientes(cliente_id),
    FOREIGN KEY (producto_id)
    REFERENCES productos(producto_id)
);

CREATE TABLE servicios_productos (
    servicio_id INT NOT NULL,
    producto_id INT NOT NULL,
    cantidad INT NOT NULL,
    PRIMARY KEY (servicio_id, producto_id),
    FOREIGN KEY (servicio_id)
    REFERENCES servicios(servicio_id),
    FOREIGN KEY (producto_id)
    REFERENCES productos(producto_id)
);
```

```sql
CREATE TABLE citas (
    cita_id INT NOT NULL AUTO_INCREMENT,
    cliente_id INT NOT NULL,
    empleado_id INT NOT NULL,
    fecha_hora DATETIME NOT NULL,
    PRIMARY KEY (cita_id),
    FOREIGN KEY (cliente_id)
    REFERENCES clientes(cliente_id),
    FOREIGN KEY (empleado_id)
    REFERENCES empleados(empleado_id)
);
```

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

1. Obtener el nombre de los empleados y la cantidad de servicios que han prestado, ordenados de mayor a menor cantidad de servicios:

   - Consulta

```
SELECT e.nombre, COUNT(s.ID) AS cantidad_servicios
FROM empleados e
LEFT JOIN servicios s ON e.ID = s.empleado_id
GROUP BY e.ID
ORDER BY cantidad_servicios DESC;
```

   - Resultado

nombre	cantidad_servicios
Diamond	1
Leisha	1
Madelaine	1
Roland	1
Bobette	1
Leila	1

2. Obtener el nombre de los clientes que tienen más de 30 años y que han comprado algún producto vendido por la barbería:

   - Consulta

```
SELECT c.nombre
FROM clientes c
JOIN ventas v ON c.ID = v.cliente_id
JOIN productos p ON v.producto_id = p.REF
WHERE c.edad > 30;
```

   - Resultado

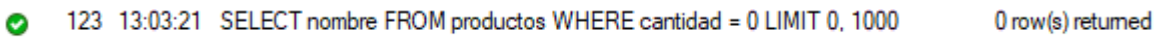
nombre
Gussie
Jonathan
Dorthea
Edmundo
Lida
Trinh
Lucile

3. Obtener el nombre de los productos vendidos por la barbería que se han agotado:

   - Consulta

```
SELECT nombre
FROM productos
WHERE cantidad = 0;
```

   - Resultado



4. Obtener el nombre del proveedor que ha suministrado la mayor cantidad de productos a la barbería:

   - Consulta

```
SELECT proveedor
FROM productos
GROUP BY proveedor
ORDER BY SUM(cantidad) DESC
LIMIT 1;
```

   - Resultado

proveedor
Fletcher

5. Obtener el nombre de los empleados y la cantidad de productos que han vendido, ordenados de mayor a menor cantidad de productos:

   - Consulta

```
SELECT e.nombre, COUNT(v.producto_id) AS cantidad_productos_vendidos
FROM empleados e
LEFT JOIN ventas v ON e.ID = v.empleado_id
GROUP BY e.ID
ORDER BY cantidad_productos_vendidos DESC;
```

   - Resultado

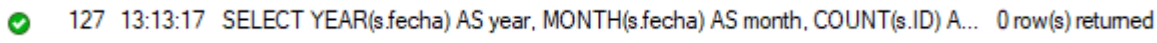
nombre	cantidad_productos_vendidos
Shila	4
Augustus	4
Taryn	4
Windy	4
Sung	4
Marsha	0
Kimbery	0

6. Obtener el número de servicios prestados por mes, para el año 2022:

   - Consulta

```
SELECT YEAR(s.fecha) AS year, MONTH(s.fecha)
AS month, COUNT(s.ID) AS cantidad_servicios
FROM servicios s
WHERE YEAR(s.fecha) = 2022
GROUP BY YEAR(s.fecha), MONTH(s.fecha)
ORDER BY year, month;
```

   - Resultado

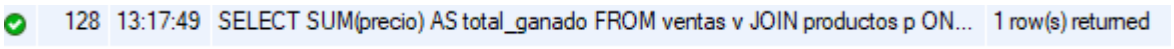


7. Obtener la cantidad total de dinero ganado por la barbería por la venta de productos, en el mes de enero de 2023:

   - Consulta

```
SELECT SUM(precio) AS total_ganado
FROM ventas v
JOIN productos p ON v.producto_id = p.REF
WHERE MONTH(v.fecha) = 1 AND YEAR(v.fecha) = 2023;
```

   - Resultado

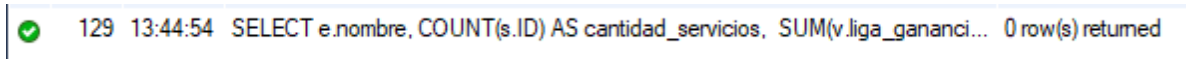


8. Obtener la cantidad de servicios prestados por cada empleado, junto con el total de dinero ganado por ellos a través de la venta de productos, en el año 2022:

   - Consulta

```sql
SELECT e.nombre, COUNT(s.ID) AS cantidad_servicios,
SUM(v.liga_ganancia) AS total_ganado
FROM empleados e
LEFT JOIN servicios s ON e.ID = s.empleado_id
LEFT JOIN ventas v ON e.ID = v.empleado_id
WHERE YEAR(s.fecha) = 2022 OR YEAR(v.fecha) = 2022
GROUP BY e.ID;
```

- Resultado



9. Obtener el número de citas por empleado, ordenados de mayor a menor número de citas, para el mes de diciembre de 2023:

- Consultado

```sql
SELECT e.nombre, COUNT(c.ID) AS cantidad_citas
FROM empleados e
JOIN citas c ON e.ID = c.empleado_id
WHERE MONTH(c.fecha_hora) = 12 AND YEAR(c.fecha_hora) = 2023
GROUP BY e.ID
ORDER BY cantidad_citas DESC;
```

- Resultado

10. La consulta SQL para obtener el nombre de los clientes que tienen más de 50 años y que han recibido servicios en la barbería sería la siguiente:

- Consulta

```sql
SELECT DISTINCT c.nombre
FROM clientes c
INNER JOIN servicios s ON c.ID = s.cliente_id
WHERE c.edad > 50;
```

- Resultado

nombre
Rory
Art
Mitzie
Lera
Jimmie
Gerard
Letitia
Stormy

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):

1. Vista de los clientes que tienen más de 50 años y que han recibido servicios:

    - Consulta

```
CREATE VIEW clientes_mayores_servicios AS
SELECT c.nombre, c.edad, s.productos_consumidos, s.fecha
FROM clientes c
INNER JOIN servicios s ON c.ID = s.cliente_id
WHERE c.edad > 50;
```

    - Resultado

nombre	edad	productos_consumidos	fecha
Rory	59	champu: Noah Lott	2023-02-17
Art	56	champu: Tad Moore	2023-02-17
Mitzie	57	champu: Xavier Breath	2023-02-17
Lera	54	champu: Hugo First	2023-02-17
Jimmie	54	champu: Bruce Easley	2023-02-17
Gerard	51	champu: Tommy Gunn	2023-02-17
Letitia	52	champu: Theresa Green	2023-02-17
Stormy	58	champu: Bess Eaton	2023-02-17

2. Vista de los empleados y sus especialidades:

    - Consulta

```
CREATE VIEW empleados_especialidades AS
SELECT e.nombre, e.especialidad
FROM empleados e;
```

- Resultado

nombre	edad	productos_consumidos	fecha
Rory	59	champu: Noah Lott	2023-02-17
Art	56	champu: Tad Moore	2023-02-17
Mitzie	57	champu: Xavier Breath	2023-02-17
Lera	54	champu: Hugo First	2023-02-17
Jimmie	54	champu: Bruce Easley	2023-02-17
Gerard	51	champu: Tommy Gunn	2023-02-17
Letitia	52	champu: Theresa Green	2023-02-17
Stormy	58	champu: Bess Eaton	2023-02-17

3. Vista de los servicios realizados por cada empleado:

- Consulta

```
CREATE VIEW servicios_empleados AS
SELECT e.nombre AS empleado, COUNT(s.ID) AS num_servicios
FROM empleados e
LEFT JOIN servicios s ON e.ID = s.empleado_id
GROUP BY e.ID;
```

- Resultado

nombre	especialidad
Karl	Mining Director
Cristie	Product Sales Orchestrator
Gregory	IT Associate
Ismael	Customer Mining Planner
Elwood	Corporate Banking Supervisor
Sandy	Hospitality Director
Chassidy	Chief Designer
Evan	Sales Facilitator
Marlon	International Architect
Jorge	Investor Strategist

4. Vista de los productos con su cantidad disponible:

- Consulta

```sql
CREATE VIEW productos_disponibles AS
SELECT p.nombre, p.cantidad
FROM productos p;
```

- Resultado

nombre	cantidad
Producto marca Phillip D. Bagg	50
Producto marca Faye Kinnitt	13
Producto marca Sandy Spring	22
Producto marca June Bugg	34
Producto marca Danielle Soloud	6
Producto marca Cody Pendant	43
Producto marca Owen Cash	60
Producto marca Skip Roper	6
Producto marca Lon Moore	20
Producto marca Douglas Furr	75

5. Vista de las ventas realizadas por cada empleado:

- Consulta

```sql
CREATE VIEW ventas_empleados AS
SELECT e.nombre AS empleado, SUM(v.liga_ganancia) AS ganancias
FROM empleados e
LEFT JOIN ventas v ON e.ID = v.empleado_id
GROUP BY e.ID;
```

- Resultado

empleado	ganancias
Karl	486.00
Cristie	310.00
Gregory	511.00
Ismael	288.00
Elwood	299.00
Sandy	567.00
Chassidy	619.00
Evan	454.00
Marlon	523.00
Jorge	520.00

6. Vista de las citas agendadas para cada empleado:

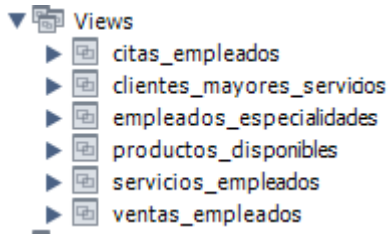
- Consulta

```sql
CREATE VIEW citas_empleados AS
SELECT e.nombre AS empleado, COUNT(c.ID) AS num_citas
FROM empleados e
LEFT JOIN citas c ON e.ID = c.empleado_id
GROUP BY e.ID;
```

- Resultado

empleado	num_citas
Karl	1
Cristie	1
Gregory	1
Ismael	1
Elwood	1
Sandy	1
Chassidy	1

- Resultados totales



Generar al menos 4 procedimientos almacenados:

1. Obtener el listado de los empleados con la cantidad de citas que han tenido durante el mes actual:

    - Consulta

```
DELIMITER //
CREATE PROCEDURE sp_citas_empleados_mes_actual()
BEGIN
  SELECT empleados.nombre, COUNT(citas.id) as citas_mes_actual
  FROM empleados
  JOIN citas ON empleados.id = citas.empleado_id
  WHERE MONTH(citas.fecha_hora) = MONTH(CURRENT_DATE())
  GROUP BY empleados.id;
END //

DELIMITER ;
```

```
CALL sp_citas_empleados_mes_actual();
```

    - Resultado

nombre	citas_mes_actual
Karl	1
Cristie	1
Gregory	1
Ismael	1
Elwood	1
Sandy	1
Chassidy	1
Evan	1

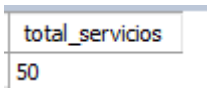
2. Obtener el número total de servicios realizados por un empleado en particular:

    - Consulta

```
DELIMITER //

CREATE PROCEDURE sp_total_servicios_empleado(IN empleado_id INT)
BEGIN
  SELECT COUNT(*) AS total_servicios
  FROM servicios
  WHERE empleado_id = empleado_id;
END //

DELIMITER ;
```

```
CALL sp_total_servicios_empleado();
```

- Resultado



total_servicios
50

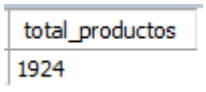
3. Obtener la cantidad de productos en existencia de un proveedor en particular:

    - Consulta

```
DELIMITER //

CREATE PROCEDURE sp_cantidad_productos_proveedor(IN proveedor VARCHAR(50))
BEGIN
  SELECT SUM(cantidad) AS total_productos
  FROM productos
  WHERE proveedor = proveedor;
END //

DELIMITER ;
```

```
CALL sp_cantidad_productos_proveedor();
```

    - Resultado



total_productos
1924

4. Obtener la lista de clientes que han comprado productos durante el mes actual y el monto total de sus compras:

    - Consulta

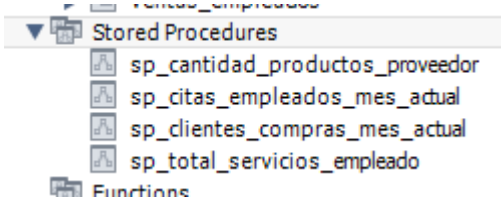
```sql
DELIMITER //

CREATE PROCEDURE sp_clientes_compras_mes_actual()
BEGIN
  SELECT clientes.nombre, SUM(ventas.liga_ganancia) AS monto_total_compras
  FROM clientes
  JOIN ventas ON clientes.id = ventas.cliente_id
  WHERE MONTH(ventas.fecha) = MONTH(CURRENT_DATE())
  GROUP BY clientes.id;
END //

DELIMITER ;
```

```sql
CALL sp_clientes_compras_mes_actual();
```

- Resultado

nombre	monto_total_compras
Enedina	486.00
Merrie	310.00
Zachery	511.00
Hiroko	288.00
Lester	299.00
Patty	567.00
Yvonne	619.00
Nolan	454.00
Deon	523.00

- Resultados totales



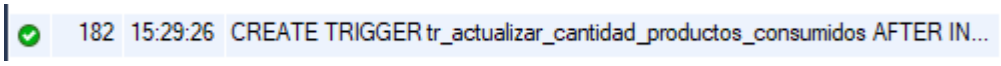
Generar al menos 4 triggers:

1. Este trigger utiliza la tabla productos y crea una tabla adicional llamada registro_productos_consumidos

   - Consulta

```
DELIMITER $$
CREATE TRIGGER tr_actualizar_cantidad_productos_consumidos
AFTER INSERT ON servicios
FOR EACH ROW
BEGIN
    DECLARE i INT DEFAULT 1;
    DECLARE id_producto INT;
    DECLARE cantidad_consumida INT;
    DECLARE nombre_producto VARCHAR(50);

    WHILE i <= LENGTH(NEW.productos_consumidos) DO
        SET id_producto = SUBSTRING_INDEX(SUBSTRING_INDEX(NEW.productos_consumidos, ',', i), ',', -1)
        SET cantidad_consumida = SUBSTRING_INDEX(SUBSTRING_INDEX(NEW.productos_consumidos, ',', i+1),
                                 SUBSTRING_INDEX(SUBSTRING_INDEX(NEW.productos_consumidos, ',', i), '
        SET nombre_producto = (SELECT nombre FROM productos WHERE REF = id_producto);

        UPDATE productos
        SET cantidad = cantidad - cantidad_consumida
        WHERE REF = id_producto;

        INSERT INTO registro_productos_consumidos(servicio_id, producto_id, cantidad_consumida, nombr
        VALUES (NEW.ID, id_producto, cantidad_consumida, nombre_producto, CURDATE());

        SET i = i + 2;
    END WHILE;
END$$
DELIMITER ;
```

   Resultado

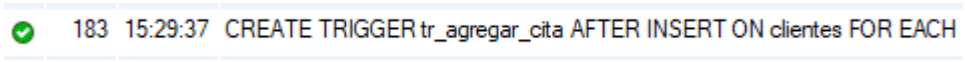


2. Trigger para agregar una cita por defecto de 30 minutos cuando se inserta un cliente en la tabla de citas:

   - Consulta

```
DELIMITER //

CREATE TRIGGER tr_agregar_cita
AFTER INSERT ON clientes
FOR EACH ROW
BEGIN
    INSERT INTO citas (cliente_id, empleado_id, fecha_hora)
    VALUES (NEW.ID, 1, DATE_ADD(NOW(), INTERVAL 30 MINUTE));
END //

DELIMITER ;

DELIMITER //
```

- Resultado

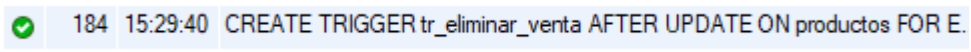


3. Trigger para eliminar una venta cuando se actualiza la cantidad a cero en la tabla de productos:

- Consulta

```
DELIMITER //

CREATE TRIGGER tr_eliminar_venta
AFTER UPDATE ON productos
FOR EACH ROW
BEGIN
    IF NEW.cantidad = 0 THEN
        DELETE FROM ventas WHERE producto_id = NEW.REF;
    END IF;
END //

DELIMITER ;
```

- Resultado

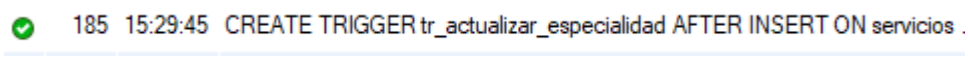


4. Trigger para actualizar la especialidad de un empleado si ha realizado un servicio de una nueva especialidad:

- Consulta

```
DELIMITER //

CREATE TRIGGER tr_actualizar_especialidad
AFTER INSERT ON servicios
FOR EACH ROW
BEGIN
    UPDATE empleados
    SET especialidad = NEW.productos_consumidos
    WHERE ID = NEW.empleado_id AND NOT EXISTS (
        SELECT * FROM servicios
        WHERE empleado_id = NEW.empleado_id
        AND productos_consumidos <> NEW.productos_consumidos
    );
END //

DELIMITER ;
```

- Resultado

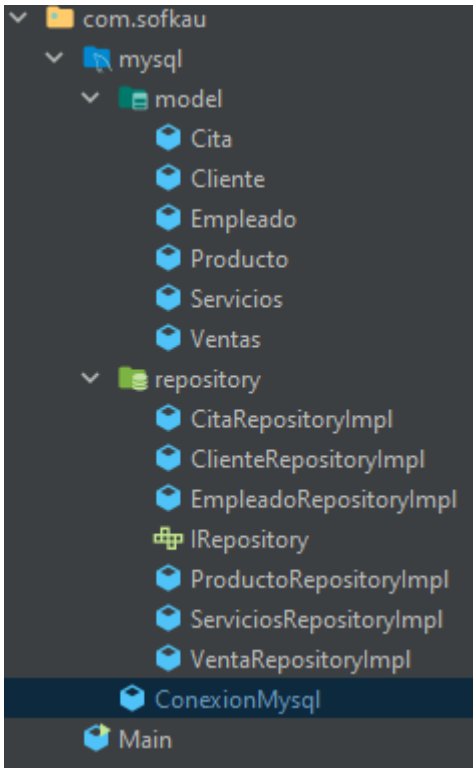


Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java:

- **Conexión mysql**

```
public class ConexionMysql {
    1 usage
    private static String url = "jdbc:mysql://localhost:3306/barberia";
    1 usage
    private static String username = "root";
    1 usage
    private static String password = "sasa";
    3 usages
    private static Connection connection;

    7 usages  Eros Jose Adarraga Jimenez *
    public static Connection getConnection() {
        if (connection == null) {
            try {
                connection = java.sql.DriverManager.getConnection(url, username, password);
            } catch (Exception e) {
                e.printStackTrace();

            }
        }
        return connection;
```

- **Creación de una clase y repositorio por cada tabla**



- **Iteración de 50 registros**

```
Faker faker = new Faker();

try (Connection conn = ConexionMysql.getConnection()) {

    IRepository<Producto> repositoryProducto = new ProductoRepositoryImpl();
    IRepository<Cliente> repositoryCliente = new ClienteRepositoryImpl();
    IRepository<Cita> repositoryCita = new CitaRepositoryImpl();
    IRepository<Empleado> repositoryEmpleado = new EmpleadoRepositoryImpl();
    IRepository<Ventas> repositoryVentas = new VentaRepositoryImpl();
    IRepository<Servicios> repositoryServicios = new ServiciosRepositoryImpl();

    System.out.println("========== insertar producto ==============");

    for (int i = 1; i <= 50; i++) {

        Empleado empleadonew = new Empleado();

        empleadonew.setCedula(String.valueOf(faker.number().numberBetween(1000000000,1999999999)));
        empleadonew.setNombre(faker.name().firstName());
        empleadonew.setEspecialidad(faker.job().title());
        repositoryEmpleado.save(empleadonew);

        Producto productonew = new Producto();
        productonew.setNombre("Producto marca "+faker.funnyName().name());
        productonew.setCantidad(faker.number().numberBetween(1,100));
        productonew.setPrecio((double) faker.number().numberBetween(200,1733));
        productonew.setProveedor(faker.name().firstName());
```

```java
Cliente clientenew = new Cliente();
clientenew.setCedula(String.valueOf(faker.number().numberBetween(1000000000,1999999999)));
clientenew.setNombre(faker.name().firstName());
clientenew.setProfesion(faker.job().title());
clientenew.setTelefono(faker.phoneNumber().subscriberNumber( length: 8));
clientenew.setCorreo(faker.internet().emailAddress());
clientenew.setDireccion(faker.address().streetAddress());
clientenew.setEdad(faker.number().numberBetween(10,60));
repositoryCliente.save(clientenew);

Cita citanew = new Cita();
citanew.setCliente(i);
citanew.setEmpleado(i);
citanew.setFechaHora(new Date());
repositoryCita.save(citanew);

Ventas ventasnew = new Ventas();
ventasnew.setFecha(new Date());
ventasnew.setClienteId(i);
ventasnew.setProductoId(i);
ventasnew.setEmpleadoId(i);
ventasnew.setLigaGanancia((double) faker.number().numberBetween(266,678));
repositoryVentas.save(ventasnew);

Servicios serviciosnew = new Servicios();
serviciosnew.setFecha(new Date());
```

- **Resultados por cada tabla**

(Citas)

ID	cliente_id	empleado_id	fecha_hora
31	31	31	2023-02-17 00:00:00
32	32	32	2023-02-17 00:00:00
33	33	33	2023-02-17 00:00:00
34	34	34	2023-02-17 00:00:00
35	35	35	2023-02-17 00:00:00
36	36	36	2023-02-17 00:00:00
37	37	37	2023-02-17 00:00:00
38	38	38	2023-02-17 00:00:00
39	39	39	2023-02-17 00:00:00
40	40	40	2023-02-17 00:00:00
41	41	41	2023-02-17 00:00:00
42	42	42	2023-02-17 00:00:00
43	43	43	2023-02-17 00:00:00
44	44	44	2023-02-17 00:00:00
45	45	45	2023-02-17 00:00:00
46	46	46	2023-02-17 00:00:00
47	47	47	2023-02-17 00:00:00
48	48	48	2023-02-17 00:00:00
49	49	49	2023-02-17 00:00:00
50	50	50	2023-02-17 00:00:00

(Clientes

ID	cedula	nombre	profesion	telefono	correo	edad	direccion
36	1989997786	Suzann	Human Accounting Associate	94412432	ashlea.marks@hotmail.com	40	0465 Jer
37	1511784802	Laci	Investor Architect	50671890	precious.gutmann@gmail.com	27	16643 Nc
38	1009646021	Karleen	Retail Executive	79368234	sydney.hudson@yahoo.com	55	73548 St
39	1172707691	Natividad	Legacy Farming Represent...	44440215	francis.koepp@gmail.com	43	284 Steu
40	1218669228	Brittaney	Lead Technician	84818849	delois.parisian@gmail.com	18	2890 Des
41	1106820303	Ciara	Construction Engineer	22385957	ty.nader@gmail.com	10	6956 Leo
42	1447066287	Hilario	Legal Orchestrator	18073273	jeramy.shields@gmail.com	38	4207 Tre
43	1947670995	Leopoldo	Regional Hospitality Designer	50725702	homer.fritsch@yahoo.com	15	08078 Ka
44	1529218582	Israel	Advertising Coordinator	35256529	cordelia.oconner@yahoo.com	15	062 Torp
45	1788798319	Veda	District Analyst	21523625	grover.powlowski@hotmail.com	23	46574 Ar
46	1951542202	Graig	National Facilitator	95804556	coleen.hackett@hotmail.com	50	132 Lock
47	1325462403	Gaston	Customer Manager	19734658	garret.wyman@yahoo.com	35	7003 Dac
48	1629808597	Gabriel	District Developer	97185345	irwin.swaniawski@gmail.com	48	40900 Sp
49	1029048798	Kitty	Human Healthcare Developer	30093297	kendrick.greenholt@hotmail.com	54	1011 Bec
50	1702708656	Brice	Design Agent	29766556	dulcie.kunde@hotmail.com	28	280 Harv

(Empleados)

ID	cedula	nombre	especialidad
37	1343857089	Jeffry	Global Real-Estate Developer
38	1348328610	Neville	Government Architect
39	1141630700	Boyce	IT Designer
40	1701130940	Bobby	Chief Coordinator
41	1823730558	Gale	Investor Designer
42	1550382407	Carlton	Ac
43	1021822135	Darin	Manufacturing Architect
44	1994798630	Shad	Future Architect
45	1853042513	Mohammed	Administration Supervisor
46	1661887380	Grayce	Internal Community-Services ...
47	1483910464	Maryjo	Sales Supervisor
48	1921137451	Lissette	Product Consultant
49	1250554438	Kip	District Manufacturing Liaison

Investor Designer

(Productos)

REF	nombre	cantidad	precio	proveedor
37	Producto marca Bo D. Satva	60	1172.00	Demetrius
38	Producto marca Helen Highwater			
39	Producto marca Iona Ford	11	1024.00	Lowell
40	Producto marca Robin Andis Merr...	25	266.00	Kristy
41	Producto marca Owen Money	23	1416.00	Daren
42	Producto marca Elmer Sklue	35	1173.00	Clemente
43	Producto marca Xavier Money	48	646.00	Rachael
44	Producto marca Rosa Shore	3	531.00	Derick
45	Producto marca Doug Love Fitzhugh	89	1072.00	Fletcher
46	Producto marca Don Thatt	38	1345.00	Hilario
47	Producto marca Mark Z. Spot	61	807.00	Rosaria
48	Producto marca Don Thatt	70	323.00	Zane
49	Producto marca Al Dente	75	500.00	Eddy

Producto marca Bo D. Satva

(Ventas)

ID	empleado_id	cliente_id	producto_id	fecha	liga_ganancia
38	38	38	38	2023-02-17	2323.00
39	39	39	39	2023-02-17	2323.00
40	40	40	40	2023-02-17	2323.00
41	41	41	41	2023-02-17	2323.00
42	42	42	42	2023-02-17	2323.00
43	43	43	43	2023-02-17	2323.00
44	44	44	44	2023-02-17	2323.00
45	45	45	45	2023-02-17	2323.00
46	46	46	46	2023-02-17	2323.00
47	47	47	47	2023-02-17	2323.00
48	48	48	48	2023-02-17	2323.00
49	49	49	49	2023-02-17	2323.00
50	50	50	50	2023-02-17	2323.00

(Servicios)

ID	empleado_id	cliente_id	productos_consumidos	duracion_procedimiento	fecha
39	39	39	champu: Bill Board	16	2023-02-17
40	40	40	champu: Rex Karrs	23	2023-02-17
41	41	41	champu: Roger Overandout	26	2023-02-17
42	42	42	champu: Brandy Bottle	12	2023-02-17
43	43	43	champu: Trina Forest	20	2023-02-17
44	44	44	champu: Rita Booke	17	2023-02-17
45	45	45	champu: Neil B. Formy	15	2023-02-17
46	46	46	champu: Hy Gene	27	2023-02-17
47	47	47	champu: Mike Rohsopht	14	2023-02-17
48	48	48	champu: Jane Linkfence	21	2023-02-17
49	49	49	champu: Al Fresco	15	2023-02-17
50	50	50	champu: Kenny Penny	15	2023-02-17

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

En cuanto a la implementación del modelo de base de datos para la barbería, considero que el modelo relacional es adecuado y cumple con los requisitos planteados en el ejercicio. La estructura de las tablas y sus relaciones permite una gestión eficiente de la información y la realización de consultas y operaciones de forma sencilla.

Sin embargo, en algunos casos una base de datos no relacional puede ser una mejor opción, por ejemplo, en situaciones en las que se requiera un almacenamiento escalable y flexible, o en sistemas que manejen grandes cantidades de datos no estructurados. En este caso, si la barbería llegara a manejar una cantidad de datos significativamente mayor, una base de datos no relacional podría ser más adecuada.

En resumen, considero que el modelo relacional utilizado en este ejercicio es adecuado para las necesidades actuales de la barbería, pero podría ser necesario considerar una base de datos no relacional en caso de que las necesidades de almacenamiento y procesamiento de datos evolucionen en el futuro.

Anexos:

- **barberia.sql**
- **consultas10.sql**
- **normalizacion_barberia.sql**
- **procedimiento_barberia.sql**
- **trigger_barberia.sql**
- **view_barberia.sql**
- **Barbería_E-R.jpeg**
- **bases.mwb**
- **Conexion-Barberia-mysql**