

use northwind;

-- Procedimeinto para crear Cliente

CREATE PROCEDURE `createCliente`

IN codigoCliente INT,

```
IN nombreCliente VARCHAR(50),  
IN NombreContacto VARCHAR(50),  
IN Direccion VARCHAR(50),  
IN ciudad VARCHAR(20),  
IN codigoPostal VARCHAR(10),  
IN pais VARCHAR(15)  
)
```

```
INSERT INTO customers (CustomerID, CustomerName, ContactName, Address, City, PostalCode,  
Country)
```

```
VALUES (codigoCliente, nombreCliente, NombreContacto, Direccion, ciudad, codigoPostal, pais);
```

```
-- Procedimiento almacenado para consultar un cliente por su código
```

```
CREATE PROCEDURE `readCliente`(  
IN codigoCliente INT
```

```
)
```

```
SELECT * FROM customers WHERE CustomerID = codigoCliente;
```

```
-- Procedimiento almacenado para modificar los detalles de un cliente
```

```
CREATE PROCEDURE `updateCliente`(  
IN codigoCliente INT,
```

```
IN nuevoNombreCliente VARCHAR(50),
```

```
IN nuevoNombreContacto VARCHAR(50),
```

```
IN nuevaDireccion VARCHAR(50),
```

```
IN nuevaCiudad VARCHAR(20),
```

```
IN nuevoCodigoPostal VARCHAR(10),
```

```
IN nuevoPais VARCHAR(15)
```

```
)
```

UPDATE customers

SET CustomerName = nuevoNombreCliente,

ContactName = nuevoNombreContacto,

Address = nuevaDireccion,

City = nuevaCiudad,

PostalCode = nuevoCodigoPostal,

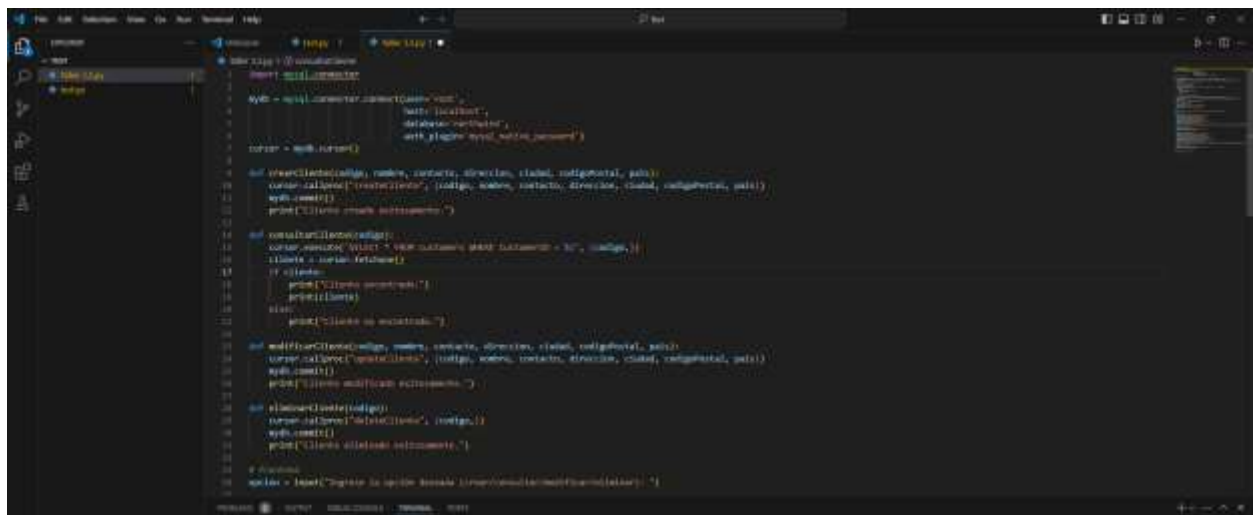
Country = nuevoPais

WHERE CustomerID = codigoCliente;

-- Procedimiento almacenado para eliminar un cliente por su código

CREATE PROCEDURE `deleteCliente`(
IN codigoCliente INT
)

DELETE FROM customers WHERE CustomerID = codigoCliente;



```
1 CREATE PROCEDURE `deleteCliente`  
2 (  
3     IN codigoCliente INT  
4 )  
5 BEGIN  
6     DELETE FROM customers WHERE CustomerID = codigoCliente;  
7  
8     SELECT * FROM customers WHERE CustomerID = codigoCliente;  
9  
10    PRINT('Cliente eliminado exitosamente');  
11  
12    IF @@ROWCOUNT = 0  
13    THEN  
14        PRINT('No se pudo eliminar el cliente');  
15    ELSE  
16        PRINT('Cliente eliminado exitosamente');  
17    END IF;  
18  
19    PRINT('Fin del procedimiento');  
20 END
```

```
1  # Import necessary modules
2  import sys
3  import os
4  import random
5  import time
6  import json
7  import pickle
8  import math
9  import logging
10 import argparse
11
12 # Define the main function
13 def main():
14     # Parse command-line arguments
15     parser = argparse.ArgumentParser()
16     parser.add_argument('--data_dir', type=str, default='./data')
17     parser.add_argument('--model_dir', type=str, default='./models')
18     parser.add_argument('--log_dir', type=str, default='./logs')
19     parser.add_argument('--seed', type=int, default=42)
20     parser.add_argument('--epochs', type=int, default=100)
21     parser.add_argument('--batch_size', type=int, default=32)
22     parser.add_argument('--num_workers', type=int, default=4)
23     parser.add_argument('--device', type=str, default='cuda:0')
24     args = parser.parse_args()
25
26     # Set the random seed
27     random.seed(args.seed)
28
29     # Load the data
30     data_loader = DataLoader(args.data_dir, args.batch_size, args.num_workers)
31
32     # Create the model
33     model = Model(args.device)
34
35     # Create the optimizer
36     optimizer = optim.Adam(model.parameters())
37
38     # Train the model
39     for epoch in range(1, args.epochs + 1):
40         # Train for one epoch
41         train_loss = 0
42         train_acc = 0
43         for batch_idx, (data, target) in enumerate(data_loader):
44             # Forward pass
45             output = model(data)
46             # Compute the loss
47             loss = criterion(output, target)
48             # Backward pass
49             loss.backward()
50             # Update the parameters
51             optimizer.step()
52             optimizer.zero_grad()
53             # Compute the accuracy
54             _, predicted = output.max(1, keepdim=True)
55             acc += predicted.eq(target).sum().item()
56             train_loss += loss.item()
57         # Print the training loss and accuracy
58         print(f'Epoch {epoch}: Loss {train_loss / len(data_loader)}, Acc {train_acc / len(data_loader)}')
```

[illegible]

```
import mysql.connector

mydb = mysql.connector.connect(user='root',
                               host='localhost',
                               database='northwind',
                               auth_plugin='mysql_native_password')

cursor = mydb.cursor()

def crearCliente(codigo, nombre, contacto, direccion, ciudad, codigoPostal,
                pais):
    cursor.callproc("createCliente", (codigo, nombre, contacto, direccion,
    ciudad, codigoPostal, pais))
    mydb.commit()
    print("Cliente creado exitosamente.")

def consultarCliente(codigo):
    cursor.execute("SELECT * FROM customers WHERE CustomerID = %s", (codigo,))
    cliente = cursor.fetchone()
    if cliente:
        print("Cliente encontrado:")
        print(cliente)
    else:
        print("Cliente no encontrado.")
```

```

def modificarCliente(codigo, nombre, contacto, direccion, ciudad, codigoPostal,
pais):
    cursor.callproc("updateCliente", (codigo, nombre, contacto, direccion,
ciudad, codigoPostal, pais))
    mydb.commit()
    print("Cliente modificado exitosamente.")

def eliminarCliente(codigo):
    cursor.callproc("deleteCliente", (codigo,))
    mydb.commit()
    print("Cliente eliminado exitosamente.")

# frontend
opcion = input("Ingrese la opción deseada (crear/consultar/modificar/eliminar):
")

if opcion == "crear":
    codigo_Cliente = input("Ingrese código cliente: ")
    nombre_Cliente = input("Ingrese nombre: ")
    contacto_Cliente = input("Ingrese contacto: ")
    direccion_Cliente = input("Ingrese dirección: ")
    ciudad_Cliente = input("Ingrese ciudad: ")
    codigoPostal_Cliente = input("Ingrese código postal: ")
    pais_Cliente = input("Ingrese país: ")
    crearCliente(codigo_Cliente, nombre_Cliente, contacto_Cliente,
direccion_Cliente, ciudad_Cliente, codigoPostal_Cliente, pais_Cliente)

elif opcion == "consultar":
    codigo_Cliente = input("Ingrese código cliente: ")
    consultarCliente(codigo_Cliente)

elif opcion == "modificar":
    codigo_Cliente = input("Ingrese código cliente: ")
    nombre_Cliente = input("Ingrese nuevo nombre: ")
    contacto_Cliente = input("Ingrese nuevo contacto: ")
    direccion_Cliente = input("Ingrese nueva dirección: ")
    ciudad_Cliente = input("Ingrese nueva ciudad: ")
    codigoPostal_Cliente = input("Ingrese nuevo código postal: ")
    pais_Cliente = input("Ingrese nuevo país: ")
    modificarCliente(codigo_Cliente, nombre_Cliente, contacto_Cliente,
direccion_Cliente, ciudad_Cliente, codigoPostal_Cliente, pais_Cliente)

elif opcion == "eliminar":
    codigo_Cliente = input("Ingrese código cliente: ")
    eliminarCliente(codigo_Cliente)

```

```
else:  
    print("Opción no válida.")
```