```kotlin
package com.example.new_clientes_sqlite

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class ClientesSQLite (context: Context) : SQLiteOpenHelper (context, DATABASE_NAME, null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_NAME = "NewClientes.db"
        private const val DATABASE_VERSION = 1
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTableQuery = """
            CREATE TABLE clientes (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nombre TEXT NOT NULL,
                dni TEXT NOT NULL
                )
                """.trimIndent()
        db.execSQL(createTableQuery)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        //Eliminar la tabla NewClientes si existe ya volver a crearla
        db.execSQL("DROP TABLE IF EXISTS clientes")
        onCreate(db)
    }



    fun insert(nuevoCliente: Cliente): Long {
        val db = getWritableDatabase()

        val values = ContentValues()
        values.put("nombre", nuevoCliente.nombre)
        values.put("dni", nuevoCliente.dni)

        val newId = db.insert("clientes", null, values)
        db.close()
        return newId
    }

    /**
     * Leer los datos de un cliente usando la clase Cliente
     * @param idCliente la ID del cliente en BBDD
```

```
50          * @return el objeto Cliente con su información
51          */
52
53
54         @SuppressLint("Range")
55         fun read(idCliente: Long) : Cliente {
56             val db = getReadableDatabase()
57             val selectQuery = "SELECT * FROM clientes WHERE id = ?"
58             val cursor: Cursor = db.rawQuery (selectQuery, arrayOf (idCliente.toString()))
59             var cliente = Cliente ("", "")
60             if (cursor.moveToFirst()) {
61                 val nombre = cursor.getString(cursor.getColumnIndex("nombre"))
62                 val dni = cursor.getString(cursor.getColumnIndex("dni"))
63                 cliente = Cliente(nombre, dni)
64             }
65             cursor.close()
66             db.close()
67             return cliente
68         }
69
70         /**
71          * Actualizar un cliente con los datos de forma individual
72          * @param idCliente ID del cliente en BBDD
73          * @param cliente objeto de clase Cliente con los datos a actualizar
74          * @return el número de filas afectadas en un Int
75          */
76
77         fun update(idCliente: Long, cliente : Cliente):Int {
78             val db = getWritableDatabase()
79             val values = ContentValues()
80             values.put ("nombre", cliente.nombre)
81             values.put ("dni", cliente.dni)
82
83             val affectedRows = db.update("clientes", values, "id = ?", arrayOf(idCliente.toString()))
84             db.close()
85             return affectedRows
86         }
87
88         /**
89          * Método que elimina un cliente de la Base de Datos
90          * @param idcliente ID del cliente en BBDD
91          * @return el número de filas afectadas en un Int
92          */
93         fun delete (idCliente:Long): Int {
94             val db = getWritableDatabase()
95             val affectedRows = db.delete("clientes", "id = ?", arrayOf(idCliente.toString()))
96             db.close()
97             return affectedRows
98         }
```

```
99
100       /**
101        * Método para obtener el número de clientes de nuestra BBDD
102        *@return el número de clientes en BBDD, si da error devuelve -1
103        */
104
105
106       @SuppressLint("Range")
107       fun getNumeroClientes(): Int {
108           val db = getReadableDatabase()
109           val selectQuery = "SELECT count (*) as numClientes FROM clientes"
110           val cursor: Cursor = db.rawQuery(selectQuery, null)
111           var num = -1
112           if (cursor.moveToFirst()) {
113               num = cursor.getInt(cursor.getColumnIndex("numClientes"))
114           }
115           cursor.close()
116           db.close()
117           return num
118       }
119
120       @SuppressLint("Range")
121       fun getListadoClientes(): List<Cliente> {
122           val clientesList = mutableListOf<Cliente>()
123           val db = getReadableDatabase()
124           val selectQuery = "SELECT * FROM clientes ORDER BY dni DESC"
125           val cursor: Cursor = db.rawQuery(selectQuery, null)
126
127           if (cursor.moveToFirst()) {
128               do {
129                   val nombre = cursor.getString(cursor.getColumnIndex("nombre"))
130                   val dni = cursor.getString(cursor.getColumnIndex("dni"))
131                   val cliente = Cliente (nombre, dni)
132                   clientesList.add(cliente)
133
134               } while (cursor.moveToNext())
135           }
136           cursor.close()
137           db.close()
138           return clientesList
139       }
140
141   }
```