MODULO 4. ACCESO A LA INFORMACIÓN DE BASE DE DATOS

LORETO PELEGRIN CASTILLO

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL
Región de Murcia
Sef
Servicio Regional DE EMPLEO y FORMACIÓN
AVANZA

1. Conexión a la Base de Datos

```
import sqlite3

# Conectarse a una base de datos SQLite

conn = sqlite3.connect('Empresa.db')

cursor = conn.cursor()
```

2. Creación de Tablas

Para crear una tabla hemos de indicar el nombre de la misma, el nombre de los atributos, el tipo de los mismos, y las restricciones a aplicar tanto a la tabla como a las columnas que la forman.

La sintaxis es SQL es:

CREATE TABLE <Nombre_Tabla> (<Atributo> <Tipo de dato>, <Atributo> <Tipo de dato>)

CREATE TABLE IF NOT EXISTS <Nombre_Tabla>: La cláusula IF NOT EXISTS se puede usar a la hora de crear tablas, esto permite que la tabla solo se cree si no existe previamente.

Como la tabla se crea de forma automática la primera vez que se ejecuta el programa,

nos evitará problemas de duplicidad.

Ejemplo en Python:

```
# Crear una tabla llamada 'T_Usuarios'

cursor.execute('''

CREATE TABLE IF NOT EXISTS T_Usuarios (

    Id INTEGER,

    Nombre TEXT,

    Email TEXT

  )

''')
```

2.1. Tipo de datos

En cuanto al tipo de dato nos encontramos con los siguientes:

SQLite	Python
NULL	None
INTEGER	int
REAL	float
TEXT	str
BLOB	bytes

2.2. Modificadores de tipo

Además de indicar el tipo de dato de cada columna, contamos con modificadores de tipo que pueden tomar los siguientes valores:

NULL	Se admiten valores nulos para ese atributo.
NOT NULL	No se admiten valores nulos.
UNIQUE	No se admiten valores repetidos.

2.3. Definición de claves

Al definir la tabla podemos indicar cual de las columnas constituye la clave principal y cuales son claves secundarias, indicando el nombre del atributo y tabla con el que se relaciona.

PRIMARY KEY

FOREING KEY <nombre> REFERENCE nombre_tabla (nombre_columna)

2.4. Ejemplo completo.

```
cursor.execute('''
    TABLE IF NOT EXISTS T_Usuarios (
        Id INTEGER NOT NULL UNIQUE PRIMARY KEY,
        Nombre TEXT,
        Email TEXT
      )
''')
```

3. Consulta de Datos

SELECT

Sirve para seleccionar datos de una o más tablas, va acompañado de FROM, que es desde cual tabla de datos se quiere coger la información.

```
# Seleccionar todos los usuarios
cursor.execute('SELECT * FROM T_Usuarios')
filas = cursor.fetchall()
for fila in filas:
   print(fila)
```

WHERE

La cláusula WHERE se utiliza para filtrar los resultados de una consulta, es decir, para seleccionar solo aquellas filas que cumplan una determinada condición.

Ejemplos:

```
SELECT * FROM T_Clientes WHERE ciudad = 'Madrid';
SELECT * FROM T_Clientes WHERE edad > 30;
```

GROUP BY

La cláusula GROUP BY se utiliza para agrupar filas según uno o más atributos.

Esto es útil para realizar cálculos agregados como SUM, COUNT, AVG, etc.

Ejemplo:

SELECT ciudad, COUNT(*) AS num_clientes FROM T_Clientes GROUP BY ciudad;

HAVING

La cláusula HAVING se utiliza para filtrar grupos después de aplicar GROUP BY. Es similar a WHERE, pero opera sobre grupos en lugar de filas individuales.

Ejemplo:

SELECT ciudad, COUNT(*) AS num_clientes FROM T_Clientes GROUP BY ciudad HAVING COUNT(*) > 10;

JOIN

La cláusula JOIN se utiliza para combinar filas de dos o más tablas basándose en una relación común.

SELECT clientes.nombre, pedidos.fecha_pedido

FROM clientes

INNER JOIN pedidos ON clientes.id = pedidos.id_cliente;

ORDER BY

La cláusula ORDER BY se utiliza para ordenar los resultados de una consulta.

4. Actualización de Datos

```
# Actualizar el email de un usuario
cursor.execute('''
   UPDATE T_Usuarios
   SET email = 'nuevo_email@ejemplo.com'
   WHERE id = 1
''')
```

5. Eliminación de Datos

```
# Eliminar un usuario
cursor.execute('DELETE FROM T_Usuarios WHERE id = 2')
```