**SQL en Python: Gestion BD (I)**

La gestión de Bases de Datos es un mundo muy amplio que no podemos abarcar en un sprint y mucho menos en un par de píldoras, pero sí es bueno por lo menos que conozcas cómo con SQL (y para el gestor Sqlite3) se pueden hacer acciones tales como:

* Insertes, Updates y Deletes en las tablas
* Crear y Tirar Tablas
* Usar vistas

**Creación de Bases de Datos y tablas**

Podemos crear nuestras propias bases de datos utilizando SQLite (que no es directamente trasladable a otros gestores) y luego dentro crearemos tablas usando ya sí sintaxis SQL. Nos importamos nuestras librerías y comenzamos a trabajar:

**import** pandas **as** pd

**import** sqlite3

Ahora haremos una conexión a una base de datos que en realidad no existe, pero esa es la forma de crearla:

connection**=**sqlite3.connect("Base\_de\_datos\_I.db")

Y ya podemos crearnos el cursor para poder interactuar con ella:

cursor\_gestion**=**connection.cursor()

Y ahora ya podemos crear una tabla siguiendo la siguiente sintaxis:

**CREATE** **TABLE** nombre\_tabla (

columna1 tipo\_de\_dato restricciones,

columna2 tipo\_de\_dato restricciones,

...

columnaN tipo\_de\_dato restricciones

)

Donde:

* **nombre\_tabla:** Es el nombre de la tabla que queremos crear.
* **columna1, columna2, ..., columnaN:** Son los nombres de las columnas de la tabla.
* **tipo\_de\_dato:** Es el tipo de dato de cada columna (por ejemplo, INT para enteros, VARCHAR o CHAR para cadenas de texto, DATE para fechas, etc.).
* **restricciones:** Son las restricciones o reglas para cada columna (opcional). Algunas restricciones comunes incluyen PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY, etc. (que corresponden a un índice único, a que no puede dejarse vacío el campo, a que no puede repetirse o a que tendrá que enlazarse con la clave en otra tabla)

Creemos una tabla sencilla de alumnos de un Master:

query\_create **=** '''

CREATE TABLE Master\_Class (

ID INT PRIMARY KEY, -- nombre tipo\_de\_dato restricción como hemos visto antes

NOMBRE TEXT NOT NULL,

EDAD INT NOT NULL,

CIUDAD CHARS(50), -- Le decimos que este campo siempre tiene 50 caracteres, y los que no llene los rellenará con espacios

NOTAS FLOAT

)

'''

cursor\_gestion.execute(query\_create)

<sqlite3.Cursor at 0x7f3f00b973b0>

Y una vez la tenemos, podemos buscarla en la tabla maestra:

query**=**'''

SELECT \*

FROM sqlite\_master

WHERE type=="table"

'''

cursor\_gestion.execute(query)

cursor\_gestion.fetchall()

[('table',

'Master\_Class',

'Master\_Class',

2,

'CREATE TABLE Master\_Class (\nID INT PRIMARY KEY, -- nombre tipo\_de\_dato restricción como hemos visto antes\nNOMBRE TEXT NOT NULL,\nEDAD INT NOT NULL,\nCIUDAD CHARS(50), -- Le decimos que este campo siempre tiene 50 caracteres, y los que no llene los rellenará con espacios\nNOTAS FLOAT\n)')]

Fíjate que almacena el nombre, y la query empleada para crearla.

**Insert**

Ahora que tenemos una tabla podemos insertar registros, modificarlos y borrarlos. En esta píldora trataremos el insert y dejaremos update y delete para la próxima.

Recordemos la sintaxis de un insert:

**INSERT** **INTO** nombre\_tabla (columna1, columna2, columna3, ...)

**VALUES** (valor1, valor2, valor3, ...)

Donde:

* **nombre\_tabla:** Nombre de la tabla donde se insertarán los datos.
* **columna1, columna2, ... :** Las columnas de la tabla en las que se insertarán los datos. No es necesario incluir todas las columnas, especialmente si algunas tienen valores predeterminados o son autoincrementables.
* **valor1, valor2, ... :** Los valores correspondientes a las columnas especificadas. Deben estar en el mismo orden que las columnas y deben ser del tipo de dato adecuado para cada columna.

Insertemos unos cuantos valores:

* Luis, 24, Madrid, 8.5
* Ana, 32, Lugo, 6.25
* Juan, 35, Bilbao, 5.55
* Nuria, 41, Alicante, 9.75

*# Usa este diccionario para no ir valor a valor:*

datos **=** {

"Luis": (24,"Madrid", 8.5),

"Ana": (32,"Lugo", 6.25),

"Juan": (35, "Bilbao", 5.55),

"Nuria": (51, "Alicante", 9.75)

}

**for** indice,(nombre,valores) **in** enumerate(datos.items()):

edad**=**valores[0]

ciudad**=**valores[1]

nota**=**valores[2]

query**=**f"INSERT INTO Master\_Class (ID,NOMBRE,EDAD,CIUDAD,NOTAS) VALUES ({indice},'{nombre}',{edad},'{ciudad}',{nota})"

cursor\_gestion.execute(query)

Y, esto es importante, ahora es necesario hacer algo que no habíamos hecho hasta ahora, y que se parece a lo que hacemos con los repos, hay que confirmarle a nuestro gestor que queremos hacer los cambios:

connection.commit()

Para comprobar que realmente se han hecho los cambios haríamos nuestra consulta y podríamos pasarlo a un DataFrame pero vaya, no tenemos nuestra función, no importa porque ya es hora de que te enseñe otra cosa:

query**=**'''

SELECT \*

FROM Master\_Class

'''

df**=**pd.read\_sql(query, connection)

df

Tabla

Descripción generada automáticamente

df.info()

Texto

Descripción generada automáticamente

connection.close()