# SQL en Python: Primeras Queries

En la sesión anterior vimos como conectarnos a una base de datos (bueno a un fichero) y sobre todo como crear un cursor y utilizarlo para consultar la base de datos. Pero se nos quedó pendiente ver cómo podíamos saber que tablas tiene una base de datos y así ya poder empezar a trabajar de forma práctica sobre SQL. Eso vamos a hacer en esta sesión. Lo primero imports y carga la base de datos:

[1]:

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

**import** sqlite3

*# Conectamos con la base de datos chinook.db*

connection **=** sqlite3.connect("data/chinook.db")

*# Obtenemos un cursor que utilizaremos para hacer las queries*

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

### Tablas y Schema

Para ver las tablas que hay en una base de datos con la que hemos establecido conexión en el caso de sqlite3:

[3]:

res**=**cursor\_bootcamp.execute("SELECT name FROM sqlite\_master WHERE type='table'")

tablas**=**[]

**for** name **in** res:

print(name[0])

tablas.append(name[0])

[3]:

albums

sqlite\_sequence

artists

customers

employees

genres

invoices

invoice\_items

media\_types

playlists

playlist\_track

tracks

sqlite\_stat1

films

Fíjate que hemos empleado una consulta (SELECT) sql sobre una tabla, que se denomina maestra, y le hemos pasado la query al gesto a través del cursor.

Ahora podríamos recorrer todas las tablas y sacar sus nombres de columnas o podríamos investigar las columnas de esa tabla maestra y de ahí obtener el modelo de datos. Recuerda que el modelo de datos de una Base de Datos relacional es el mapa de todas las tablas de esa Base de Datos con el nombre de sus campos, el tipo de valores que guardan esos campos y las relaciones que hay entre las tablas.

### Modelo de datos

Antes de empezar a atacar una base de datos, tendremos que saber qué hay dentro, y para ello lo mejor es ver cómo es su **modelo de datos**. Como he comentado podríamos intentar sacarlo de los nombres de los campos de las tablas o bien utilizar otros módulos externos o herramientas externas, pero dado que nosotros seremos principalmente consumidores, lo más efectivo será preguntar por él. En nuestro caso (la base Chinook de ventas de musica onlie) este es el modelo:

Escala de tiempo

Descripción generada automáticamente con confianza media

### Primeras queries con SELECT

De la sesión de introducción recordarás que la sintaxis básica de una sentencia o query SELECT tiene esta pinta:

**SELECT** campo1, campo2, campo3...

**FROM** tabla

**WHERE** condiciones

**ORDER** **BY** campo1, campo2 (**DESC**)

**LIMIT** num\_filas

Por supuesto, hay más sentencias propias de SQL que iremos viendo a lo largo de la unidad, y del bootcamp. Fíjate que las **palabras reservadas en SQL se suelen poner en mayúsculas**, para diferenciarlas del resto. No da error si se pone de otra manera, ya que SQL **no es case sensitive**, pero sí se suele hacer así.

En lo que queda de sesión vamos a hacer nuestras primeras queries, muy sencillas, para que vayas familiarizándote y ya en los ejercicios tendrás mucho donde practicar.

#### SELECT \*

Ya la hicimos sobre la tabla employees en la sesión anterior probemos ahora con otras tablas

[4]:

query**=**'''

SELECT\*

FROM tracks

'''

Como puedes ver se suele utilizar la triple comilla y la consulta en varias líneas según la parte del SELECT que corresponda, pero no es imperativo, puedes poner toda la sentencia en una sola línea tal y como hemos hecho al consultar el nombre de todas las tablas contenidas en la base de datos. Completemos la ejecución y volquemos a un DataFrame tal y como vimos:

[5]:

cursor\_bootcamp.execute(query)

resultado**=**cursor\_bootcamp.fetchall()

columnas**=**[d[0] **for** d **in** cursor\_bootcamp.description]

df**=**pd.DataFrame(resultado,columns**=**columnas)

df

[5]:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

### Selección de campos

Seleccionemos ahora algunos campos únicamente y además cambiémosle el nombre al vuelo, mediante la sintaxis campo as nuevo\_nombre. **Si quieres poner espacios en el nombre del campo, tendrás que rodear el string con comillas dobles**

Y recuerda: SQL no es sensible a mayusculas y minusculas. ("este\_CampO" == "ESTE\_CAMPO")

[6]:

query**=**'''

SELECT name AS "Nombre Cancion", composer AS "Compositor"

FROM tracks

'''

cursor\_bootcamp.execute(query)

resultado**=**cursor\_bootcamp.fetchall()

columnas**=**[d[0] **for** d **in** cursor\_bootcamp.description]

df**=**pd.DataFrame(resultado,columns**=**columnas)

df

[6]:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

### LIMIT y DISTINCT

Para terminar la sesión veamos cómo usar dos modificadores LIMIT (que ya vimos en la introducción teórica) y DISTINCT (que es el equivalente del método unique en pandas)

#### LIMIT

Se usa para acotar el número de registros de la query. Va siempre al final. Por ejemplo LIMIT 10

[8]:

query='''

SELECT name AS "Nombre Cancion"

FROM tracks

LIMIT 10

'''

cursor\_bootcamp.execute(query)

resultado=cursor\_bootcamp.fetchall()

columna=[d[0] for d in cursor\_bootcamp.description] # es lo mismo que: columna="Nombre Cancion"

df=pd.DataFrame(resultado, columns=columna)

df

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza media

#### DISTINCT

Se usa para obtener todos los registros únicos, es decir, sin duplicados. Lo podremos emplear para eliminar dupicados (aunque te recomiendo que no modifiques los datos de las bases de datos, vuelcalo a pandas y modifica ahí), como para ver todas las casuísticas de un campo en concreto.

**Mucho cuidado con esta sentencia ya que si la tabla tiene miles o millones de registros, puede ralentizar mucho la query.**

[9]:

Selection deleted

query**=**'''

SELECT DISTINCT Composer

FROM tracks

'''

cursor\_bootcamp.execute(query)

resultado**=**cursor\_bootcamp.fetchall()

columna**=**[d[0] **for** d **in** cursor\_bootcamp.description]

df**=**pd.DataFrame(resultado,columns**=**columna)

df

[9]:

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente