Database Programming using Python

Python supports several databases. The most commonly used are:

- **SQLite** (built-in, file-based)
- **MySQL / MariaDB**
- **PostgreSQL**
- **MongoDB** (NoSQL)
- **SQLAlchemy** (ORM for SQL databases)

In this Lecture we will learn how to work with **SQLite** using Python's built-in `sqlite3` module.

1. Using sqlite3 module for Database operations

- **sqlite3** is a built-in Python module that provides an interface to SQLite databases.
- SQLite is a lightweight, file-based database that doesn't require a separate server process.
- sqlite3 provides a straightforward and simple-to-use interface for interacting with SQLite databases.
- There is no need to install this module separately as it comes along with Python after the 2.5x version.
- Used in Mobile apps, browsers, small desktop tools (e.g., Android, Firefox, Python apps)

NOTE

- SQLite is just like other relational database management systems (RDBMS) like MySQL or PostgreSQL.
- Just that it doesn't require a separate server process like MySQL and PostgreSQL
- We don't need to install anything additional to get started because Python has built-in support for SQLite through the sqlite3 module.

SQLite Vs sqlite3 Vs SQL

- SQL => A language used to interact with relational databases.
- SQLite => A lightweight relational database engine just like MySQL.
- sqlite3 => A Python module to work with SQLite databases.

Step 1: Importing sqlite3 module

In [8]:
```
import sqlite3
```

Step 2: Connecting to a Database

In [19]:
```
# To interact with an SQLite database, we first need to establish a connection to it using the connec
# If the specified database file doesn't exist, SQLite will create it automatically.
conn = sqlite3.connect("decodeaiml_db.db")
```

Step 3: Creating a Table

In [20]:
```
cursor.execute('DROP TABLE IF EXISTS Employees')
```

Out[20]:
```
<sqlite3.Cursor at 0x270738a0240>
```

In [21]:
```
# After establishing the connection, we need to create a cursor object to execute SQL queries on the 
cursor = conn.cursor()
```

In [23]:
```
cursor.execute("""CREATE TABLE IF NOT EXISTS Employees(
        id int,
        name text,
        experience int,
        location text
    )
    """)
```

Out[23]:
```
<sqlite3.Cursor at 0x270738a0a40>
```

Step 4: Inserting Data

In [24]:
```
cursor.execute("""INSERT INTO Employees VALUES
            (1,'Sanjeev',6, "India"),
            (2,'Shambahvi',5, "India"),
            (3,'Aman',3, "Remote"),
            (4,'Anjali',3, "Remote")
        """)
conn.commit()
```

Step 5: Querying Data

In [25]:
```
# 1. List all employees with details
result = cursor.execute("select * from Employees")
for row in result:
    print(row)
```

```
(1, 'Sanjeev', 6, 'India')
(2, 'Shambahvi', 5, 'India')
(3, 'Aman', 3, 'Remote')
(4, 'Anjali', 3, 'Remote')
```

In [32]:
```
# 2. List all employees with details sorted in descending order
result = cursor.execute("SELECT * FROM Employees ORDER BY Experience")
for row in result:
    print(row)
```

```
(3, 'Aman', 3, 'Remote')
(4, 'Anjali', 3, 'Remote')
(2, 'Shambahvi', 5, 'India')
(1, 'Sanjeev', 6, 'India')
```

In [35]:
```
# 2. List all employees with location Remote
result = cursor.execute("SELECT * FROM Employees WHERE Location = 'Remote'")
for row in result:
    print(row)
```

```
(3, 'Aman', 3, 'Remote')
(4, 'Anjali', 3, 'Remote')
```

Usecase: Reading from a Database and writing to a CSV/Excel File

In [ ]:
```
conn = sqlite3.connect("decodeaiml_db.db")
cursor = conn.cursor()
```

In [36]:
```
result = cursor.execute("select * from Employees")
employee_list = []
for row in result:
    employee_list.append(row)
```

In [37]:
```
employee_list
```

Out[37]:
```
[(1, 'Sanjeev', 6, 'India'),
 (2, 'Shambahvi', 5, 'India'),
 (3, 'Aman', 3, 'Remote'),
 (4, 'Anjali', 3, 'Remote')]
```

In [39]:
```
# import pandas and create dataframe
import pandas as pd

df = pd.DataFrame(employee_list, columns=['Id','Name','Experience','Location'])
```

In [40]:
```
df.head()
```

Out[40]:

	Id	Name	Experience	Location
0	1	Sanjeev	6	India
1	2	Shambahvi	5	India
2	3	Aman	3	Remote
3	4	Anjali	3	Remote

In [45]:
```
# Save to CSV
df.to_csv('employees_csv.csv', index=False)
```

In [43]:
```
!pip install openpyxl
```

```
Collecting openpyxl
  Downloading openpyxl-3.1.5-py2.py3-none-any.whl.metadata (2.5 kB)
Collecting et-xmlfile (from openpyxl)
  Downloading et_xmlfile-2.0.0-py3-none-any.whl.metadata (2.7 kB)
Downloading openpyxl-3.1.5-py2.py3-none-any.whl (250 kB)
Downloading et_xmlfile-2.0.0-py3-none-any.whl (18 kB)
Installing collected packages: et-xmlfile, openpyxl

   ---------------------- ---------------------- 1/2 [openpyxl]
   ---------------------- ---------------------- 1/2 [openpyxl]
   ---------------------- ---------------------- 1/2 [openpyxl]
   ---------------------- ---------------------- 1/2 [openpyxl]
   ---------------------------------------- 2/2 [openpyxl]

Successfully installed et-xmlfile-2.0.0 openpyxl-3.1.5
```

In [46]:
```
# Save to Excel
df.to_excel('employees_excel.xlsx', index=False)
```