**1. SQLite (Built-in)**

* **Library**: sqlite3
* **Use Case**: SQLite is a lightweight, file-based database engine, and the sqlite3 library is part of Python’s standard library. It's ideal for small applications or for prototyping.

**import sqlite3**

**# Connect to SQLite database (or create it)**

**conn = sqlite3.connect('example.db')**

**# Create a cursor object**

**cursor = conn.cursor()**

**# Create table**

**cursor.execute('''CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, age INTEGER)''')**

**# Insert a row of data**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Alice', 25)")**

**# Commit and close**

**conn.commit()**

**conn.close()**

**2. MySQL**

* Library: mysql-connector-python, PyMySQL
* Use Case: These libraries allow Python to interface with MySQL databases. mysql-connector-python is developed by MySQL, while PyMySQL is a third-party pure Python implementation.
* Install: pip install mysql-connector-python or pip install **pymysql**

**import mysql.connector**

**# Connect to MySQL database**

**conn = mysql.connector.connect(**

**host='localhost',**

**user='root',**

**password='password',**

**database='testdb'**

**)**

**cursor = conn.cursor()**

**# Execute query**

**cursor.execute("SELECT \* FROM users")**

**# Fetch results**

**results = cursor.fetchall()**

**for row in results:**

**print(row)**

**conn.close()**

**3. PostgreSQL**

* Library: psycopg2, asyncpg
* Use Case: psycopg2 is the most popular PostgreSQL adapter for Python. It supports transactions, efficient data handling, and more. asyncpg is a faster, asynchronous library for working with PostgreSQL.

**import psycopg2**

**# Connect to PostgreSQL database**

**conn = psycopg2.connect(**

**host="localhost",**

**database="testdb",**

**user="postgres",**

**password="password"**

**)**

**cursor = conn.cursor()**

**# Execute query**

**cursor.execute("SELECT \* FROM users")**

**# Fetch results**

**rows = cursor.fetchall()**

**for row in rows:**

**print(row)**

**conn.close()**

**Here are examples of SQL queries like CREATE, UPDATE, DELETE, etc., executed with SQLite, PostgreSQL, and MySQL using Python.**

**1. SQLite**

Setup: Using the built-in sqlite3 library.

import sqlite3

**# Connect to SQLite database (creates the file if not exists)**

**conn = sqlite3.connect('example.db')**

**cursor = conn.cursor()**

**# 1. CREATE TABLE**

**cursor.execute('''CREATE TABLE IF NOT EXISTS users (**

**id INTEGER PRIMARY KEY AUTOINCREMENT,**

**name TEXT NOT NULL,**

**age INTEGER)''')**

**# 2. INSERT DATA**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Alice', 25)")**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Bob', 30)")**

**# 3. SELECT DATA**

**cursor.execute("SELECT \* FROM users")**

**rows = cursor.fetchall()**

**for row in rows:**

**print(row)**

**# 4. UPDATE DATA**

**cursor.execute("UPDATE users SET age = 26 WHERE name = 'Alice'")**

**# 5. DELETE DATA**

**cursor.execute("DELETE FROM users WHERE name = 'Bob'")**

**# Commit changes and close**

**conn.commit()**

**conn.close()**

**2. PostgreSQL**

Setup: Using psycopg2 library.

**import psycopg2**

**# Connect to PostgreSQL database**

**conn = psycopg2.connect(**

**host="localhost",**

**database="testdb",**

**user="postgres",**

**password="password"**

**)**

**cursor = conn.cursor()**

**# 1. CREATE TABLE**

**cursor.execute('''CREATE TABLE IF NOT EXISTS users (**

**id SERIAL PRIMARY KEY,**

**name VARCHAR(100),**

**age INT)''')**

**# 2. INSERT DATA**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Alice', 25)")**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Bob', 30)")**

**# 3. SELECT DATA**

**cursor.execute("SELECT \* FROM users")**

**rows = cursor.fetchall()**

**for row in rows:**

**print(row)**

**# 4. UPDATE DATA**

**cursor.execute("UPDATE users SET age = 26 WHERE name = 'Alice'")**

**# 5. DELETE DATA**

**cursor.execute("DELETE FROM users WHERE name = 'Bob'")**

**# Commit changes and close**

**conn.commit()**

**conn.close()**

**3. MySQL**

Setup: Using mysql-connector-python library.

**import mysql.connector**

**# Connect to MySQL database**

**conn = mysql.connector.connect(**

**host="localhost",**

**user="root",**

**password="password",**

**database="testdb"**

**)**

**cursor = conn.cursor()**

**# 1. CREATE TABLE**

**cursor.execute('''CREATE TABLE IF NOT EXISTS users (**

**id INT AUTO\_INCREMENT PRIMARY KEY,**

**name VARCHAR(255),**

**age INT)''')**

**# 2. INSERT DATA**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Alice', 25)")**

**cursor.execute("INSERT INTO users (name, age) VALUES ('Bob', 30)")**

**# 3. SELECT DATA**

**cursor.execute("SELECT \* FROM users")**

**rows = cursor.fetchall()**

**for row in rows:**

**print(row)**

**# 4. UPDATE DATA**

**cursor.execute("UPDATE users SET age = 26 WHERE name = 'Alice'")**

**# 5. DELETE DATA**

**cursor.execute("DELETE FROM users WHERE name = 'Bob'")**

**# Commit changes and close**

**conn.commit()**

**conn.close()**