Până acum, Alex a învățat cum să folosească SQL pentru a gestiona datele direct în baza de date MySQL. Totuși, în aplicațiile din lumea reală, de obicei nu lucrăm direct cu baza de date – în schimb, folosim **un limbaj de programare** pentru a comunica cu baza de date.

Python, fiind unul dintre cele mai populare limbaje pentru analiza și prelucrarea datelor, permite o conectare ușoară la bazele de date MySQL și executarea de interogări SQL direct din cod. Această integrare aduce **automatizare, gestionarea dinamică a datelor și flexibilitate** în lucrul cu bazele de date.

Ce vom învăța în această lecție?

- Cum să stabilim conexiunea cu baza de date MySQL din Python.
- Ce biblioteci Python folosim pentru a lucra cu bazele de date MySQL.
- Cum să executăm interogări SQL direct din Python.

Prin combinarea **SQL-ului cu Python**, Alex va putea:

- să introducă, să actualizeze și să șteargă automat datele din bază;
- să extragă datele din bază și să le folosească în aplicațiile sale Python;
- să creeze aplicații interactive care comunică cu baza de date.

Sunteți gata? Să începem cu conectarea Python-ului și MySQL-ului!

Instalarea driverului MySQL

Pentru ca Python să poată comunica cu baza de date MySQL, avem nevoie de **un driver special** – o bibliotecă care permite conectarea și schimbul de date între Python și serverul MySQL.

Alex va folosi MySQL Connector, una dintre cele mai populare biblioteci pentru lucrul cu bazele de date MySQL din Python.

Cum se instalează MySQL Connector?

Instalarea bibliotecii **mysql-connector-python** se poate face simplu folosind managerul de pachete **pip**. În linia de comandă (terminal), executați următoarea comandă:

```
pip install mysql-connector-python
```

Ce permite această bibliotecă?

- Conectarea la baza de date MySQL din Python.
- Executarea de interogări SQL (SELECT, INSERT, UPDATE, DELETE).
- Accesarea și manipularea datelor într-un mod dinamic.

După instalarea cu succes, Alex este gata să stabilească **prima conexiune cu baza de date**!

Conectarea la baza de date MySQL din Python

Acum că Alex a instalat **MySQL Connector**, următorul pas este să stabilească o conexiune cu baza de date pentru a putea comunica cu aceasta și executa interogări SQL.

Cum stabilim conexiunea cu baza de date MySQL?

Pentru conectare, folosim funcția `mysql.connector.connect()`, căreia îi transmitem parametrii necesari:

- `host` – locația bazei de date (local: "localhost"; pentru serverul la distanță: adresa IP);

- **`port`** – numărul portului (implicit 3306, poate fi omis dacă utilizăm portul standard);
- **`user`** – numele de utilizator pentru accesul la baza de date MySQL;
- **`password`** – parola pentru accesul la baza de date MySQL;
- **`database`** – numele bazei de date cu care dorim să lucrăm.

Alex dorește să se conecteze la baza sa de date library, unde va putea ulterior să execute interogări și să manipuleze datele. Să vedem cum a realizat conexiunea!

Exemplu: Cum s-a conectat Alex la baza de date?

```
import mysql.connector

# Creating a connection to the MySQL database
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="root",
  database="library"
)

# Checking if the connection was successfully established
if(mydb == None):
  print("There is no connection to database.")
else:
  print("Connection to database is created.")
```

Notă:

- Dacă folosim portul implicit MySQL (**3306**), nu este necesar să îl specificăm în conexiune.
- Metoda `is_connected()` verifică dacă conexiunea este activă.

- Dacă conexiunea nu este stabilită, trebuie verificat **dacă serverul MySQL este pornit și dacă datele de autentificare sunt corecte**.

Acum, Alex poate executa interogări SQL direct din Python!

Cum verificăm ce port folosește MySQL?

Pentru ca Alex să fie sigur că folosește **portul corect pentru conectare**, trebuie să verifice pe ce port rulează serverul său MySQL.

Există două modalități simple de a face acest lucru:

1. Verificarea portului prin interogare SQL

Dacă Alex este deja în **MySQL Monitor** sau folosește **MySQL Workbench**, poate executa simplu următoarea interogare:

```
SHOW VARIABLES LIKE 'port';
```

Rezultatul va arăta astfel:

Variable_name	Value
port	3306

Tabelul 24.1. ServerParameters

Portul implicit MySQL este **3306**, dar în funcție de configurarea serverului, acesta poate fi diferit.

2. Verificarea portului în fișierul de configurare MySQL

Dacă Alex dorește să consulte **fișierul de configurare al serverului MySQL**, îl poate găsi la următoarele locații:

Windows:

C:\ProgramData\MySQL\MySQL Server X.X\my.ini

Linux / Mac:

/etc/mysql/my.cnf sau */etc/my.cnf*

Apoi, în fișierul de configurare trebuie să găsească secțiunea [mysqld] și linia care începe cu **port**:

```
[mysqld]
port=3306
```

Dacă Alex dorește să schimbe portul, poate actualiza simplu această valoare și poate reporni serverul MySQL.

Tutorial video: 24.1. Conectarea la serverul MySQL dintr-un program Python

Mini activitate: Să stabilim conexiunile noastre!

Sarcină: Pe calculatoarele noastre, să creăm un script Python care se conectează la baza de date.

Memento: Să nu uităm ...

- să instalăm **MySQL Connector** dacă nu am făcut-o deja (`pip install mysql-connector-python`);
- să verificăm ce port folosește MySQL și să ajustăm stringul de conexiune dacă este necesar.

Concluzia lecţiei

În această lecție, Alex a învățat cum să stabilească o conexiune între**o aplicație Python și baza de date MySQL**. Acesta este primul pas esențial pentru gestionarea dinamică a datelor, deoarece permite aplicațiilor Python să comunice direct cu baza de date fără a mai fi nevoie de introducerea manuală a comenzilor SQL.

Am învăţat:

- cum să folosim biblioteca **mysql-connector-python** pentru a ne conecta la o bază de date MySQL;
- cum să configurăm corect conexiunea folosind parametrii **host, user, password și database**;
- când este necesar să specificăm **portul** și cum să îl găsim în configurația MySQL.

Acum că Alex știe cum să stabilească conexiunea cu baza de date, următorul pas logic este **executarea interogărilor SQL din Python**!

Următorul pas: Executarea interogărilor SQL din Python

În lecția următoare, Alex va învăța:

- **să introducă noi date** în baza de date MySQL folosind interogarea **INSERT**;
- **să preia date** din baza de date folosind interogarea **SELECT**;
- **să actualizeze înregistrările existente** folosind interogarea **UPDATE**;
- **să șteargă date** folosind interogarea **DELETE**.

Aceste operațiuni – **Create, Read, Update, Delete (CRUD)** –

constituie baza lucrului cu bazele de date și îi vor permite lui Alex să gestioneze complet datele din Python.

Să facem următorul pas spre integrarea completă a Python-ului și SQL-ului!