Python Programming
Unit 04 – Lecture 05 Notes
DB-API 2.0, CRUD Operations, MongoDB Overview

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Python connects to relational databases using the DB-API 2.0 pattern:

- open a connection,
- create a cursor,
- execute SQL queries,
- fetch results,
- commit changes (for INSERT/UPDATE/DELETE),
- close connection.

This lecture uses SQLite (`sqlite3`) for hands-on CRUD and discusses MongoDB at a high level.

2 Core Concepts

2.1 DB-API 2.0 Workflow (SQLite Example)

```
import sqlite3

con = sqlite3.connect("app.db")
cur = con.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS students(id INTEGER PRIMARY KEY, name TEXT)")
con.commit()

cur.execute("SELECT * FROM students")
rows = cur.fetchall()

con.close()
```

2.2 CRUD in SQL

- **Create:** `INSERT INTO table (...) VALUES (...)`
- **Read:** `SELECT ... FROM table WHERE ...`
- **Update:** `UPDATE table SET ... WHERE ...`
- **Delete:** `DELETE FROM table WHERE ...`

2.3 Parameterised Queries

Never build SQL by concatenation with user input:

```
# BAD (do not do this)
cur.execute("SELECT * FROM students WHERE sapid = '" + sapid + "'")
```

Use placeholders:

```
cur.execute("SELECT * FROM students WHERE sapid = ?", (sapid,))
```

Benefits:

- avoids SQL injection,
- handles quoting automatically,
- reduces bugs.

2.4 Fetching Results

Common methods:

- `fetchone()` for a single row
- `fetchall()` for all rows

2.5 MongoDB Overview

MongoDB is a document database:

- data is stored as documents (JSON-like),
- documents are grouped into collections.

Python driver: `pymongo` (external install). Example (conceptual):

```
# pip install pymongo
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017")
db = client["college"]
students = db["students"]
students.insert_one({"name": "Asha", "sapid": "5001", "marks": 88})
```

3 Demo Walkthrough

SQLite CRUD: `demo/sqlite_crud_demo.py`
MongoDB (optional): `demo/mongodb_optional_demo.py`

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: purpose of `cursor.execute(...)`?
Answer: It sends an SQL statement (or command) to the database engine to be executed.

Checkpoint 2 Solution

Question: why parameterised queries are safer?
Answer: They separate SQL code from data values, preventing injection and handling escaping correctly.

5 Practice Exercises (with Solutions)

Exercise 1: Update Marks

Task: Update marks for a student by SAP ID.
Solution:

```
cur.execute("UPDATE students SET marks = ? WHERE sapid = ?", (marks, sapid))
con.commit()
```

Exercise 2: Delete a Student

Task: Delete a row by SAP ID.

Solution:

```
cur.execute("DELETE FROM students WHERE sapid = ?", (sapid,))
con.commit()
```

6 Exit Question (with Solution)

Question: SQL query to update marks by SAP ID?

Answer (pattern):

```
UPDATE students SET marks = ? WHERE sapid = ?;
```