Python Programming
Unit 04 – Lecture 04 Notes
Tkinter + Databases (Concepts and Integration)

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview **1**

2 Core Concepts **2**
2.1 Relational Databases (SQL) 2
2.2 NoSQL Databases 2
2.3 SQLite for Learning and Small Apps 2
2.4 GUI-to-Database Workflow 2
2.5 Parameterised Queries (Important) 2

3 Demo Walkthrough **3**

4 Interactive Checkpoints (with Solutions) **3**

5 Practice Exercises (with Solutions) **3**

6 Exit Question (with Solution) **4**

1 Lecture Overview

Many GUI apps need to store data persistently: users, tasks, marks, inventory, etc. Files can work for simple cases, but databases provide:

- structured storage,
- fast searching,
- constraints (unique IDs),
- and transactions for safe updates.

2 Core Concepts

2.1 Relational Databases (SQL)

Relational databases store data in **tables**. Each table has:

- columns (fields), e.g., `name`, `sapid`
- rows (records), one row per student

Primary key: a column (or set of columns) that uniquely identifies a row. Examples: `id`, `sapid`.

2.2 NoSQL Databases

NoSQL databases store data in different formats depending on the type:

- document (MongoDB),
- key-value (Redis),
- graph (Neo4j),
- column-family (Cassandra).

They often offer flexible schemas and scale for certain use-cases.

2.3 SQLite for Learning and Small Apps

SQLite is an embedded relational database:

- no server is required,
- database is stored as a single file,
- Python provides `sqlite3` in the standard library.

2.4 GUI-to-Database Workflow

Typical workflow:

1. user enters values in Entry widgets,
2. program validates input,
3. program executes SQL (INSERT/SELECT/UPDATE/DELETE),
4. program refreshes UI (Listbox/Table) to show latest data.

2.5 Parameterised Queries (Important)

Never build SQL by string concatenation with user input (SQL injection risk). Use parameters:

```
cur.execute("INSERT INTO students(name, sapid) VALUES (?, ?)", (name, sapid))
```

3 Demo Walkthrough

File: `demo/tkinter_sqlite_student_app.py`

Key things to observe:

- Create the database file in `data/` automatically.
- Create table using `CREATE TABLE IF NOT EXISTS`.
- Insert rows using parameterised queries.
- Refresh a Listbox with the latest records.
- Commit changes after INSERT.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: One advantage of databases over text files?

Answer (examples): fast queries/search, data integrity via constraints, structured schema, transactions.

Checkpoint 2 Solution

Question: What is a primary key?

Answer: A primary key uniquely identifies each row and prevents duplicates for that key value.

5 Practice Exercises (with Solutions)

Exercise 1: Create a Table

Task: Write SQL to create a table `students` with columns `id` (primary key) and `name`.

Solution:

```
CREATE TABLE IF NOT EXISTS students (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL
);
```

Exercise 2: Insert a Student Safely

Task: Insert (name, sapid) using parameters.

Solution:

```
cur.execute(
    "INSERT INTO students(name, sapid) VALUES (?, ?)",
    (name, sapid)
)
con.commit()
```

6 Exit Question (with Solution)

Question: SQL to create `students(id, name)` with `id` as primary key?

Answer:

```
CREATE TABLE students (
  id INTEGER PRIMARY KEY,
  name TEXT
);
```