Loading Credentials & the Final Secured Script

AiCE Software Engineering Foundations — MySQL Programming

Your credentials are sitting safely in .env, ignored by Git. Now let's write the Python code that actually reads them — finishing the job of keeping every secret out of your source code entirely.

1. Loading Credentials in Python

Create a file named main.py, and connect to the database using the values in .env:

```
import os
import mysql.connector
from dotenv import load_dotenv

load_dotenv()

conn = mysql.connector.connect(
    host=os.getenv("DB_HOST"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database=os.getenv("DB_NAME")
)
cursor = conn.cursor()

print("Connected to aice_books successfully!")
```

load_dotenv() reads the .env file and makes its values available through os.getenv(). Notice main.py no longer contains a single real credential — someone could read the whole file and learn nothing about your actual database password.

2. Running It

```
python3 main.py
```

You should see the same "Connected to aice_books successfully!" message as before. Nothing about the behavior changed — only where the credentials live.

3. The Complete Secured CRUD Script

Now let's update main.py to perform the entire CRUD operation, exactly as in the previous unit, but fully secured:

```
import os
import mysql.connector
from dotenv import load_dotenv

load_dotenv()

conn = mysql.connector.connect(
    host=os.getenv("DB_HOST"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database=os.getenv("DB_NAME")
)
cursor = conn.cursor()

# READ
cursor.execute("SELECT * FROM books")
for row in cursor.fetchall():
    print(row)

# CREATE
cursor.execute(
    "INSERT INTO books (title, author, genre, price, quantity,
date_added) VALUES (%s, %s, %s, %s, %s, %s)",
    ("Path to Nigerian Freedom", "Obafemi Awolowo",
"Africanism", 14.50, 20, "2025-03-01")
```

```
)
cursor.execute(
    "INSERT INTO books (title, author, genre, price, quantity,
date_added) VALUES (%s, %s, %s, %s, %s, %s)",
    ("Africa Must Unite", "Kwame Nkrumah", "Africanism",
17.25, 25, "2025-01-22")
)
cursor.execute(
    "INSERT INTO books (title, author, genre, price, quantity,
date_added) VALUES (%s, %s, %s, %s, %s, %s)",
    ("Long Walk to Freedom", "Nelson Mandela", "Africanism",
21.00, 18, "2026-08-05")
)
conn.commit()

# UPDATE
cursor.execute(
    "UPDATE books SET quantity = %s WHERE title = %s",
    (35, "Africa Must Unite")
)
conn.commit()

# DELETE
cursor.execute(
    "DELETE FROM books WHERE title = %s",
    ("Africa Must Unite",)
)
conn.commit()

cursor.close()
conn.close()
```

4. Running & Confirming

```
python3 main.py
```

Then go to the MySQL terminal and confirm everything reflects automatically:

```
SELECT * FROM books;
```

You can see the same books reflected there — the CRUD operations still work exactly as before. The only thing that changed is that main.py no longer carries a real password inside it.

5. Pushing to GitHub Safely

Push your work to GitHub. Since .env is listed in .gitignore, Git skips it automatically — only main.py, requirements.txt, and .gitignore go up. Your database password never leaves your machine.

> **The Habit That Matters Most**
> *os.getenv("DB_PASSWORD") and password="your_password" both connect to the same database, identically. The only difference is who can see the password — and that difference is the entire reason this unit exists.*

Key Takeaways

You installed python-dotenv, created a .env file to hold your real database credentials, and added .env to .gitignore alongside venv/. load_dotenv() reads that file, and os.getenv() pulls each value into your connection by name — so main.py now keeps secrets where they belong, and behaves identically to every version before it. This closes out the MySQL Programming section: you can design a database and table, perform every CRUD operation directly in MySQL, drive those same operations from Python, and do it all without a single credential ever reaching GitHub — the exact standard every real-world application is expected to meet.