# LABORATORY EXERCISE 1

# DATABASE CREATION & FOUNDATIONAL DDL

**Learning Objectives**

By the end of this laboratory exercise, students should be able to:

* Differentiate between DDL (Data Definition Language) and DML (Data Manipulation Language) and master **CREATE**, **ALTER**, **DROP**, and **USE**.
* Articulate a database schema's purpose as a data storage blueprint.
* Enforce data integrity at the schema level by:
  + Utilizing appropriate data types (e.g., **INT**, **VARCHAR**, **DATE**) ensures a valid data format.
  + Defining Primary Keys to guarantee uniqueness for each record in a table.
  + Implementing Foreign Keys to enforce referential integrity between tables.
* Understand and apply the **AUTO\_INCREMENT** attribute to automate unique identifier generation.

**Prerequisite student experiences and knowledge**

Before starting this exercise, students should have:

* Understanding core database terms:
  + Database
  + Table
  + Row (record)
  + Column (attribute/field).
* Basic familiarity with entities (e.g., "Student", "Course") and their attributes.
* Conceptual understanding of how tables can be related (e.g., one-to-many, many-to-many).
* Awareness of an SQL statement and how to execute it against a database server.

**Background**

The schema is the fundamental structure of a relational database. It defines how data is organized, related, and constrained. Data Definition Language (DDL) is the subset of SQL used to build and modify this schema. A well-designed schema is not an afterthought; it is the primary factor ensuring data integrity (accuracy and consistency of data) and performance.

* **CREATE DATABASE**
  + Instantiates a new, empty database on the server. This container contains all your tables, views, and other objects.
* **CREATE TABLE**
  + Defines a new table within the selected database. This is where you specify column names, their data types, and constraints (like `PRIMARY KEY`, `NOT NULL`).
* **ALTER TABLE**
  + Used to modify an existing table's structure (e.g., adding a new column, dropping a constraint, modifying a data type).
* **DROP TABLE/ DROP DATABASE**
  + Permanently deletes the table or database and all of its data. This command should be used with extreme caution.

**Materials/Resources**

* **dbGate**
* **Personal Computer with Internet Access**
* **XAMPP/WAMP/LAMP server installed**

**Laboratory Activity**

1. **Establish Connection and Create Script**
   1. Open your database client (dbGate).
   2. Establish a connection to your local MySQL server instance using your credentials (username: often **root**, and password).
   3. Create a new SQL script file. This is typically a blank file with a **.sql** extension where you will write your commands.
2. **Create and Select the Database**
   1. In your new script, write the following SQL command. Replace **Lastname** with your actual last name.

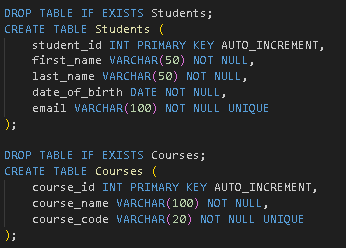
**CREATE DATABASE DB\_Lastname;**

* 1. To perform any operations within your new database, you must first select it. Write the **USE** statement next.

**USE DB\_Lastname;**

1. **Create the Tables with Detailed DDL**

Write the CREATE TABLE statements below the USE statement in the same script.

****

**Key Enhancements and Explanations:**

* **IF EXISTS**
  + Safely handles the command if the object does not exist.
* **NOT NULL** Constraint
  + Explicitly enforces that a column must have a value. This is a critical data integrity rule. A student record without a **last\_name** or **date\_of\_birth** is invalid.
* **UNIQUE** Constraint
  + Ensures all values in a column are different. This is perfect for **email** and **course\_code**, as duplicates should not be allowed in these fields.
* Indentation and Readability
  + Formatting the SQL clearly makes the table structure easy to read and debug.

1. **Execute and Verify**
   1. Execute the entire script. In most clients, you can press **Ctrl+Enter** (or **Cmd+Enter** on Mac) or click a "Run" or "Execute" button.
   2. Verification
      * Navigate to the Object Explorer or Schema Browser panel in dbGate.
      * Refresh the view if necessary. You should see your **DB\_Lastname** database listed.
      * Expand it to confirm that both the **Students** and **Courses** tables were created successfully.
      * You can further expand the tables to see their columns.

Output / Results

* A screenshot of the dbGate interface. The Object Explorer on the left shows a tree structure. The tree is expanded to show:
  + **Databases > DB\_Lastname > Tables > Students**
  + **Databases > DB\_Lastname > Tables > Courses**
  + The columns of the **Students** table are visible below it, listing **student\_id**, **first\_name**, **last\_name**, **date\_of\_birth**, and **email** with their data types and key icons (a key icon next to **student\_id**).

**QUESTIONS:**

1. What is the purpose of the AUTO\_INCREMENT attribute?

The purpose of the **AUTO\_INCREMENT** attribute is to give each new row its own number.

For example, adding the first student, MySQL gave student\_id 1, and the next student got 2 automatically. I did not type the IDs and this stopped duplicate IDs.

1. Why is VARCHAR more suitable than CHAR for names and email addresses?

VARCHAR more suitable than CHAR for names and email addresses because names and emails have different lengths.

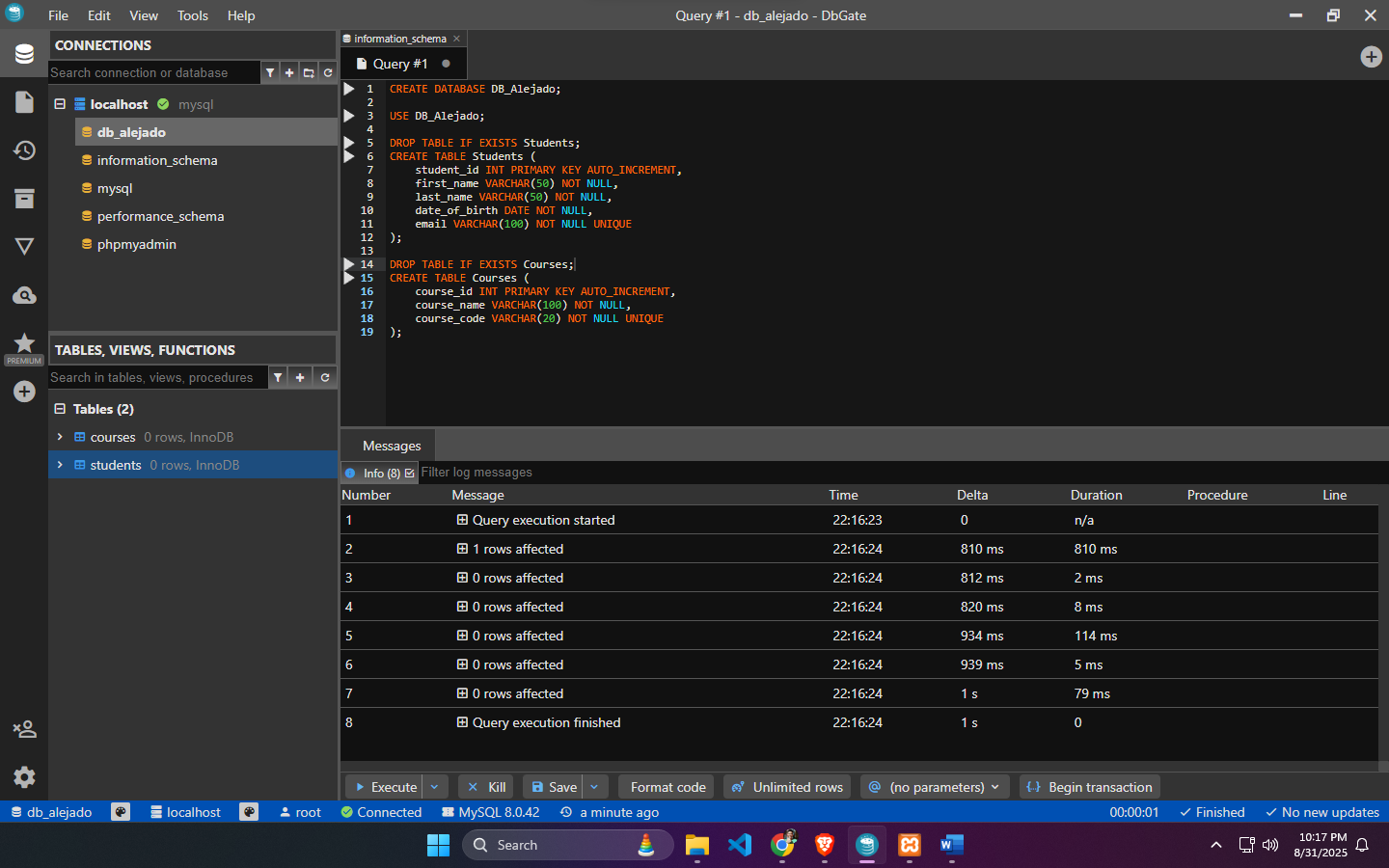
For example, “Ana” used 3 characters but “Topson” used 6, and VARCHAR saved space for both. CHAR would use the full size every time, which wastes space and can slow things.

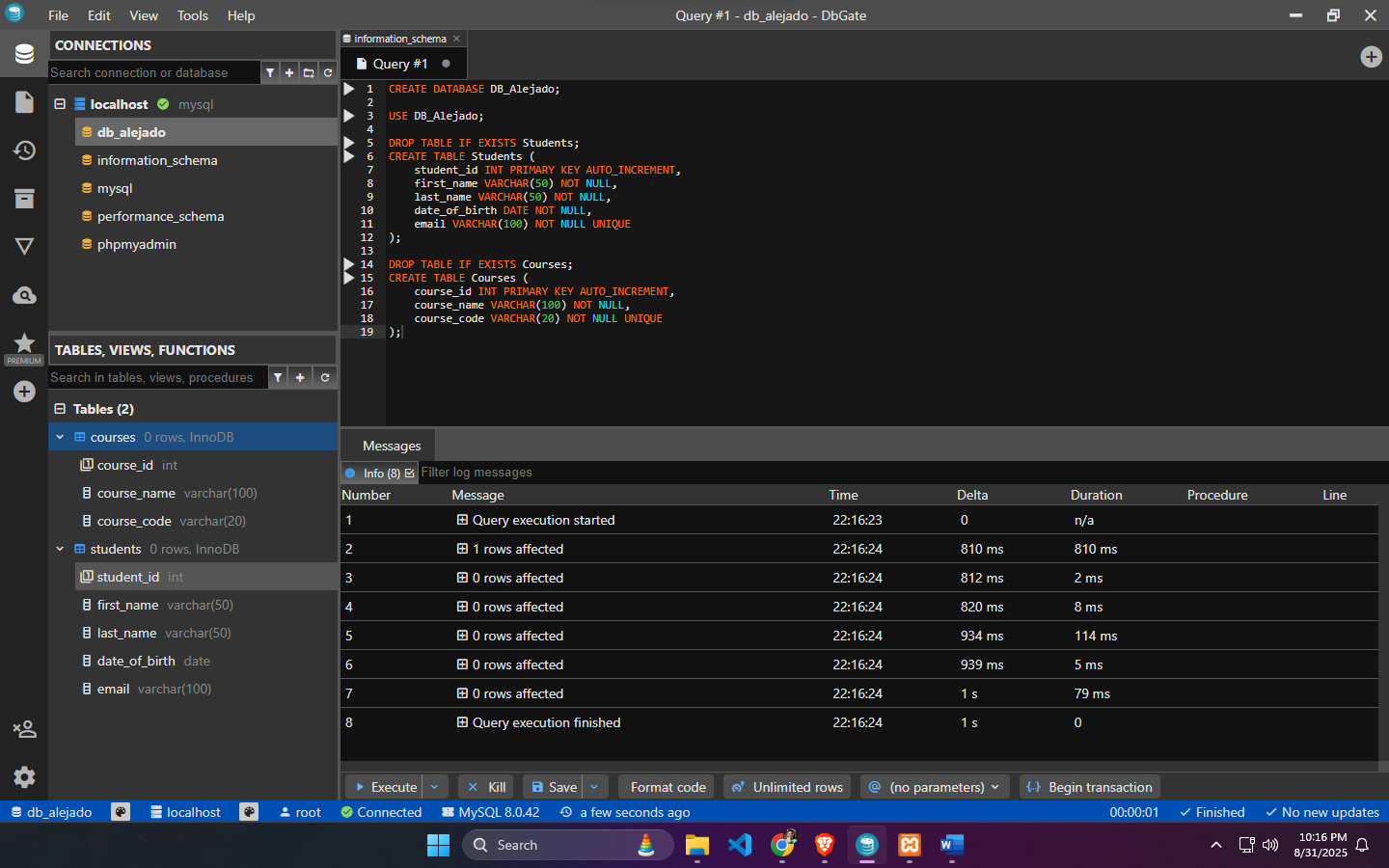
1. What potential issue is there with using only a student\_id as a primary key from a user perspective?

Potential issue is there with using only a student\_id as a primary key from a user perspective are hard for me to remember because I only remember the name not id.

For example, I could not find a student easily if I only knew their name and not their ID. So I keep student\_id as the primary key, but I also search by name or unique email for users.

**Output / Results**





**Conclusion**

In conclusion, thanks to this Laboratory Exercise 1, I learned the difference between **DDL** and **DML** and I used **CREATE**, and **USE** to control my database. I now see the database schema as my blueprint for how data is stored and connected.

I kept data clean by choosing the right data types, setting Primary Keys for unique rows, and adding Foreign Keys to link tables. I also used **AUTO\_INCREMENT** instudent\_id and course\_id**,** so each new record gets its own ID automatically.