**Module 4 – Introduction to DBMS**

*Introduction to SQL*

**Theory Questions**:

1. What is SQL, and why is it essential in database management?

* Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values.
* It is the easiest way to store, update, remove, search, or retrieve information on a database.

2. Explain the difference between DBMS and RDBMS.

* The main differences are: RDBMS stores data in the form of tables, whereas DBMS stores data in the form of files. Single users are supported by DBMS, whereas multiple users are supported by RDBMS. Client-server architecture is not supported by DBMS, although it is supported by RDBMS.

3. Describe the role of SQL in managing relational databases.

* Data retrieval: SQL allows users to retrieve data from a database by filtering and selecting specific data based on criteria. This is useful for generating reports and making data-driven decisions.
* Data manipulation: SQL can be used to insert, update, and delete records in a database.
* Database structure: SQL can be used to create and modify database structures, including tables.
* Access permissions: SQL can be used to manage access permissions to a database.
* Data transformation: SQL can be used to transform raw data into useful information by joining different tables.

4. What are the key features of SQL?

* Flexibility and Scalability:

SQL offers users flexibility and scalability for relational database management systems. With SQL, it is easier to create new tables while dropping or deleting previously-created or scantily used tables.

* A Comprehensive Application Development Tool:

Programmers use SQL to program applications to access a database, making it a comprehensive and effective application development tool. SQL is suitable for every large or small organization, no matter the size.

* Rich Transactional Support

It has rich transactional support. Structured Query Language’s programming capability of handling large records while also managing several other transactions is top-notch.

* High Performance

Another feature of SQL is that it offers a high-performance programming capability for high usage, incredibly transactional, and heavy workload database systems. Structured Query Language’s programming provides different ways to describe data more analytically.

* High Availability

SQL is compatible with other databases such as Microsoft SQL Server, Oracle Database, MS Access, MySQL, SAP Adaptive Server, and more.

These RDBMSs support SQL, and it is easier to create application extensions for procedural programming and several other SQL functions that are extra features, hence making SQL a strong tool.

* High Security

SQL also has high security as one of its notable features. It’s easy to give permissions on views, procedures, and tables. So with SQL, you get optimum security for your data.

* SQL’s Management Ease

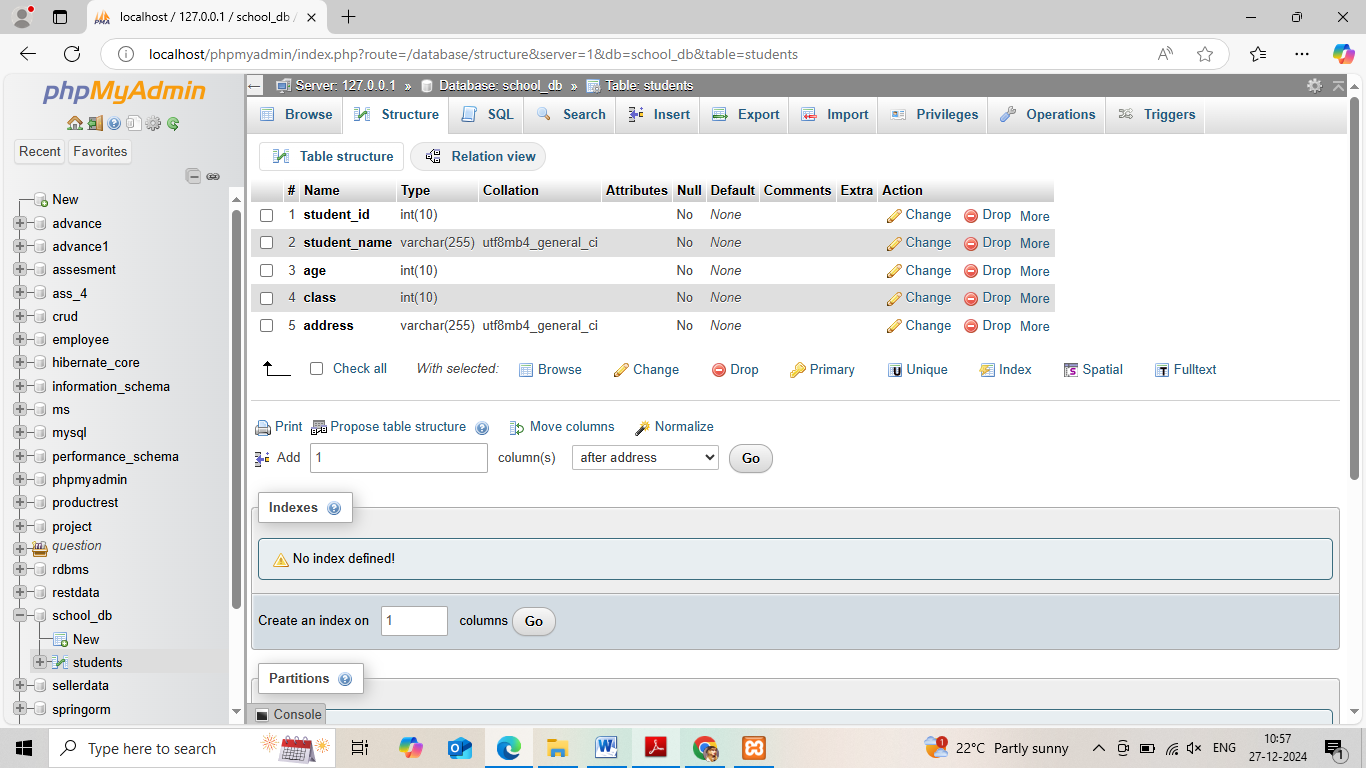
Almost every Relational Database Management System uses Structured Query Language. Some of the common and standard SQL commands include “Delete,” “Insert,” “Select,” “Update,” and “Drop.” These commands help users manage large amounts of data from a database efficiently and quickly.

* Open Source

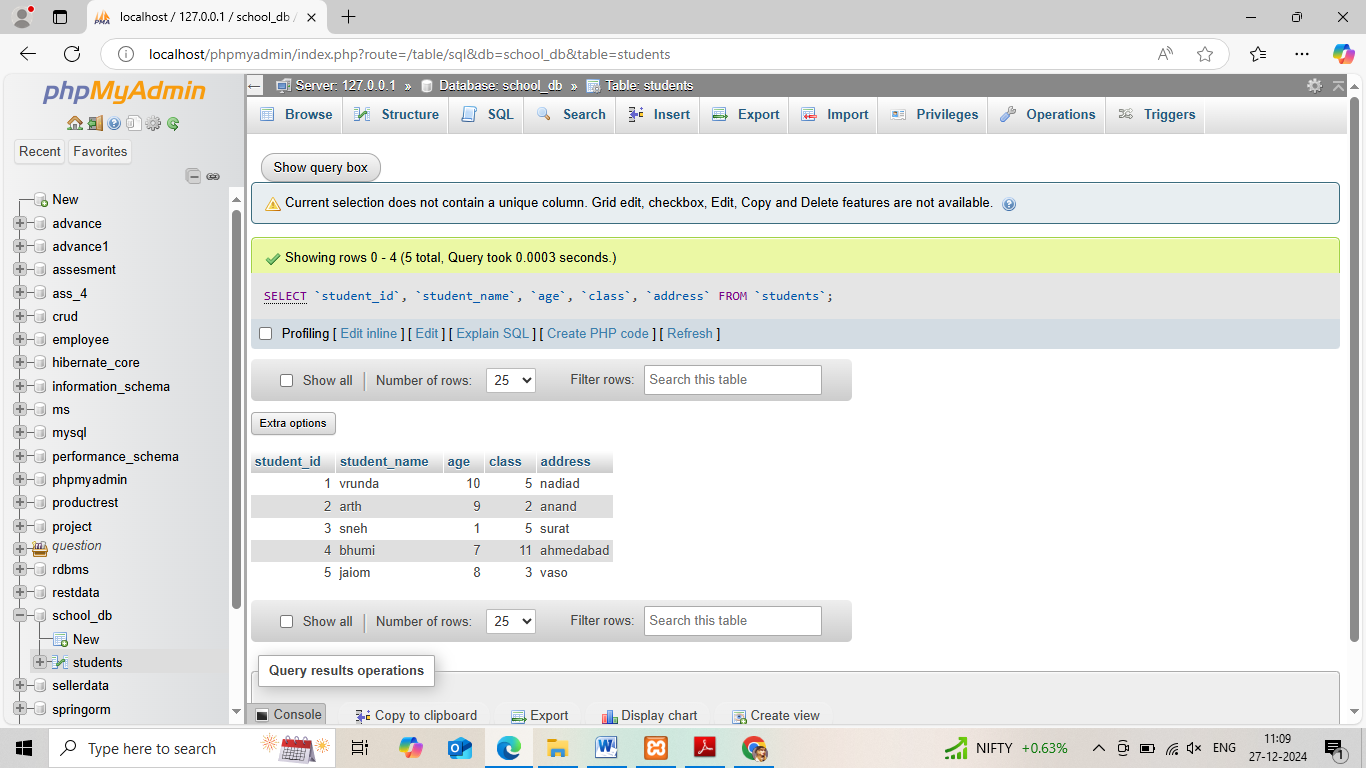
Structured Query Language has the feature of being an open-source programming language great for building relational database management systems (RDBMS). This makes it a great pick for developers and programmers who are looking for a community of professionals to learn from off, and it’s also one of the benefits of SQL.

**LAB EXERCISES**:

**Lab 1**: Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.



**Lab 2**: Insert five records into the students table and retrieve all records using the SELECT statement.



*2. SQL Syntax*

**Theory Questions**:

1. What are the basic components of SQL syntax?

* **Databases:**Databases are structured collections of data organized into tables, rows, and columns. [Databases](https://www.geeksforgeeks.org/what-is-database/)serve as repositories for storing information efficiently and provide a way to manage and access data.
* **Tables:**Tables are the fundamental building blocks of a database, consisting of rows (records) and columns (attributes or fields). Tables ensure data integrity and consistency by defining the structure and relationships of the stored information.
* **Queries:** Queries are SQL commands used to interact with databases. They enable users to retrieve, update, insert, or delete data from tables, allowing for efficient data manipulation and retrieval.
* **Constraints:**Constraints are rules applied to tables to maintain data integrity. [Constraints](https://www.geeksforgeeks.org/sql-constraints/)define conditions that data must meet to be stored in the database, ensuring accuracy and consistency.

2. Write the general structure of an SQL SELECT statement.

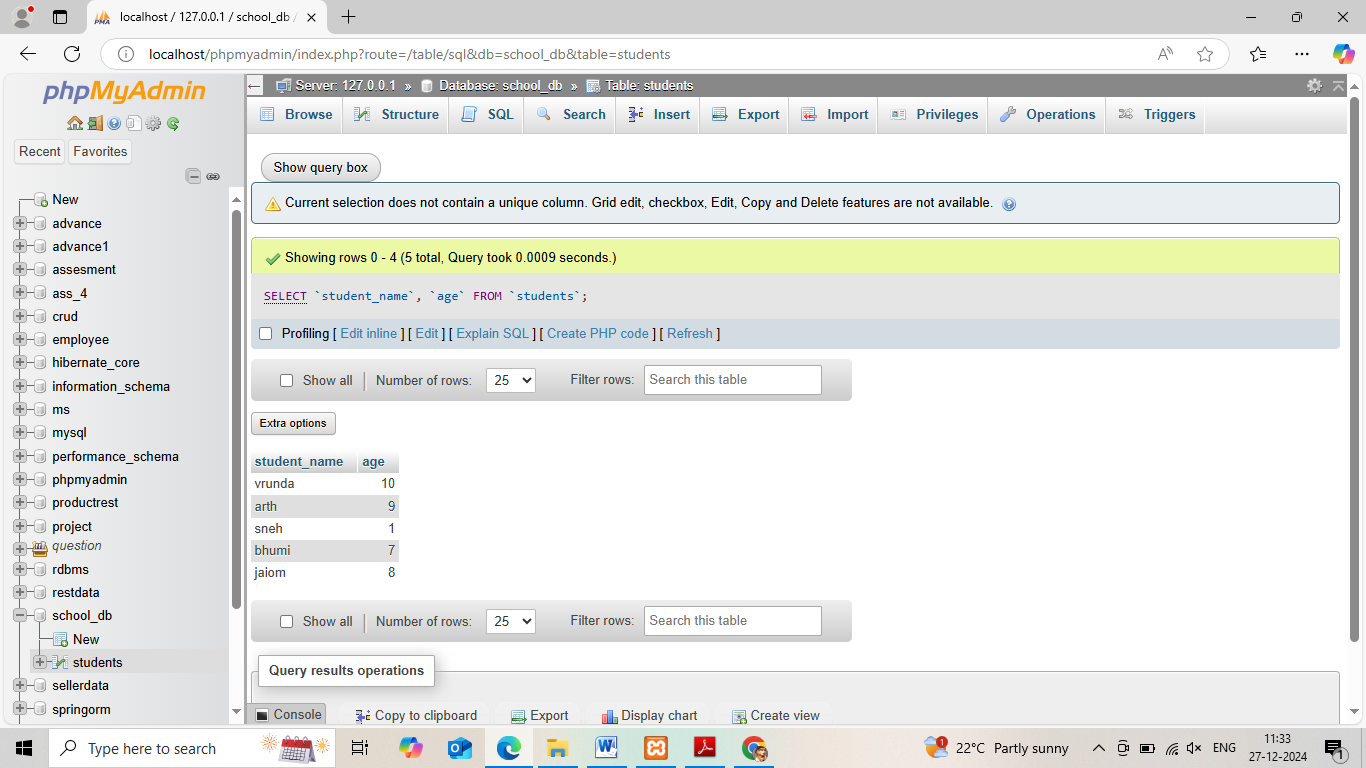
* **SELECT**: The first clause, which specifies the columns to retrieve
* **FROM**: The clause that indicates the table where the columns are located
* **WHERE**: An optional clause that specifies conditions for filtering data
* **GROUP BY**: An optional clause that can be used for grouping data
* **HAVING**: An optional clause that can be used for specifying conditions
* **ORDER BY**: An optional clause that can be used for ordering data

3. Explain the role of clauses in SQL statements.

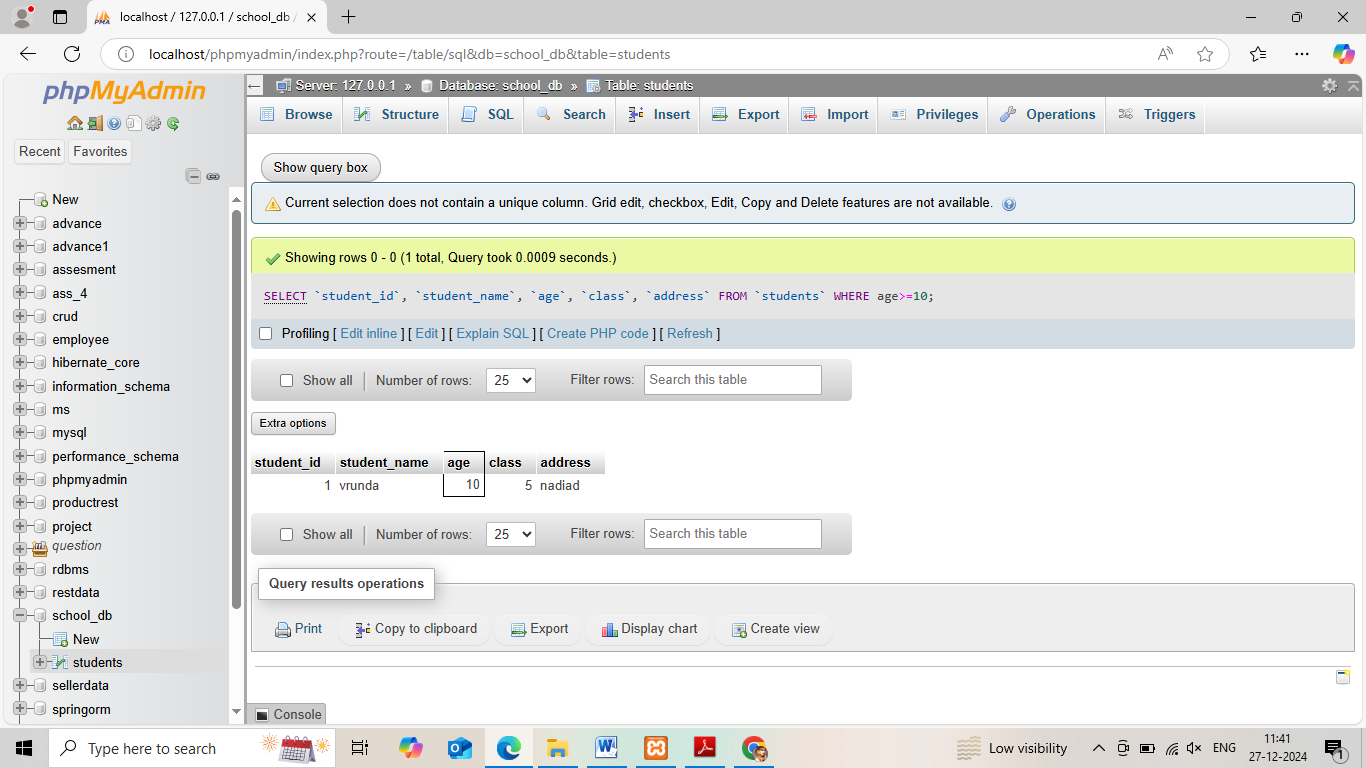
* The sql clauses can help filter out the data according to the users' needs.
* The main clauses are SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, INSERT, UPDATE, DELETE, and JOIN. Each clause has a syntax and its own set of rules and options. They can also be used in combination to create complex queries.

**LAB EXERCISES**:

**Lab 1**: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.



**Lab 2**: Write SQL queries to retrieve all students whose age is greater than 10.



*3. SQL Constraints*

**Theory Questions**:

1. What are constraints in SQL? List and explain the different types of constraints.

* Constraints in SQL are rules that ensure data accuracy, validity, and consistency. They help prevent the input of inaccurate or inconsistent data into the database.

Here are some types of constraints in SQL:

* **Primary key**: A unique constraint that requires every value in a column to be unique.
* **Foreign key**: Creates a relationship between two tables, known as a parent-child relationship. The child table foreign key must have a corresponding entry in the parent primary key column.
* **Unique**: Ensures that no duplicate values can be inserted into a column or combination of columns.
* **NOT NULL**: A constraint that tells a column that it can't have any null values in it.
* **Informational**: A constraint attribute that can be used by the SQL compiler to improve data access.

2. How do PRIMARY KEY and FOREIGN KEY constraints differ?

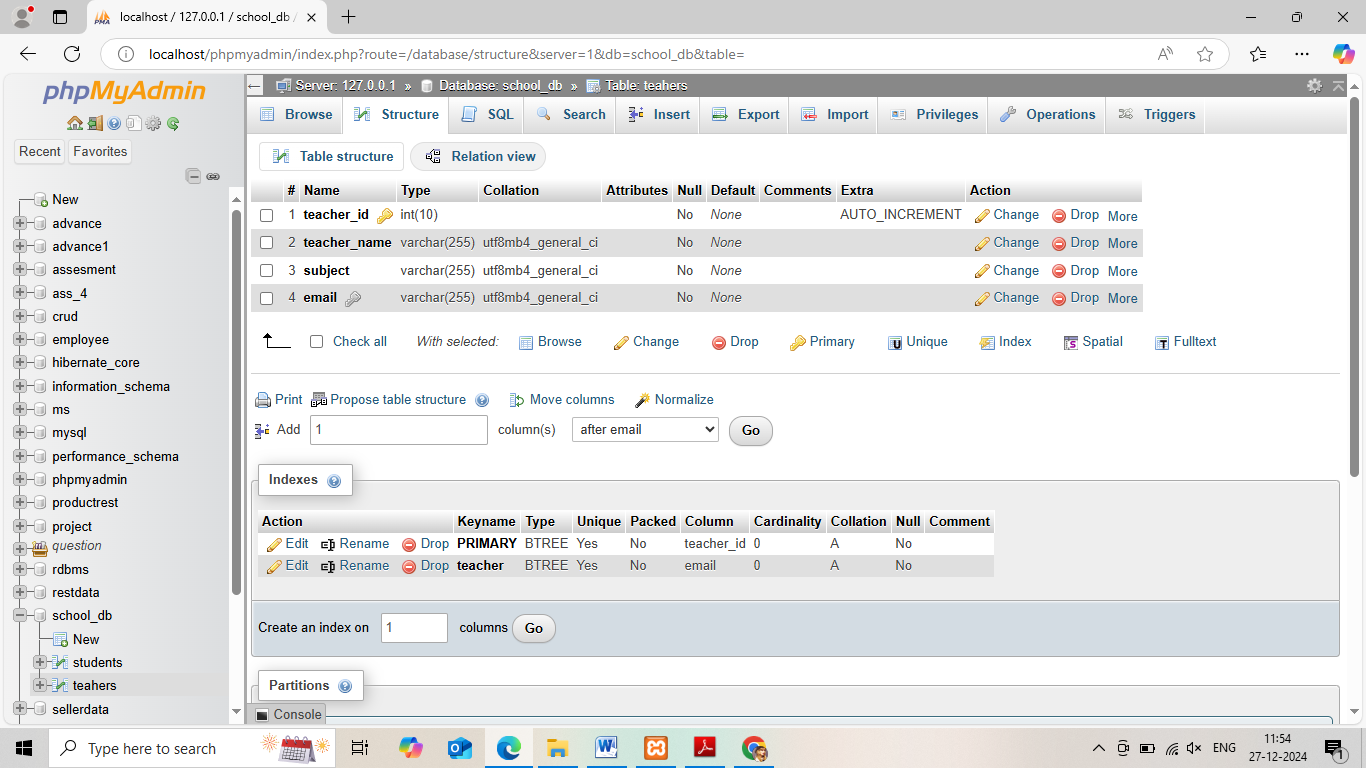
* The primary key is a unique identifier within its table, whereas a foreign key is a reference in one table to a primary key in another.
* Primary keys enforce uniqueness within their table, ensuring each record is identifiable. Foreign keys, however, are used to establish and navigate relationships between tables.

3. What is the role of NOT NULL and UNIQUE constraints?

* A NOT NULL constraint is a rule that prevents null values from being entered into one or more columns within a table.
* A unique constraint (also referred to as a unique key constraint) is a rule that forbids duplicate values in one or more columns within a table.

**LAB EXERCISES**:

 **Lab 1**: Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).



*4. Main SQL Commands and Sub-commands (DDL)*

**Theory Questions**:

1. Define the SQL Data Definition Language (DDL).

* Data definition language (DDL) describes the portion of SQL that creates, alters, and deletes database objects.

2. Explain the CREATE command and its syntax.

* The CREATE command is a Data Definition Language (DDL) command in SQL that is used to create database objects like tables, databases, and triggers.
* **Create a database**

The syntax for creating a database is CREATE DATABASE database\_name. For example, to create a database called "Books", you would use the command CREATE DATABASE Books.

* **Create a table**

The syntax for creating a table is CREATE TABLE table\_name ( column\_Name1 data\_type ( size of the column ), column\_Name2 data\_type ( size of the column), ... column\_NameN data\_type ( size of the column ) ). For example, to create a table called "Student" with five columns, you would use a command similar to CREATE TABLE Student ( ... ).

3. What is the purpose of specifying data types and constraints during table creation?

* Specifying data types and constraints when creating a table ensures the accuracy and reliability of the data in the table:
* **Data types**

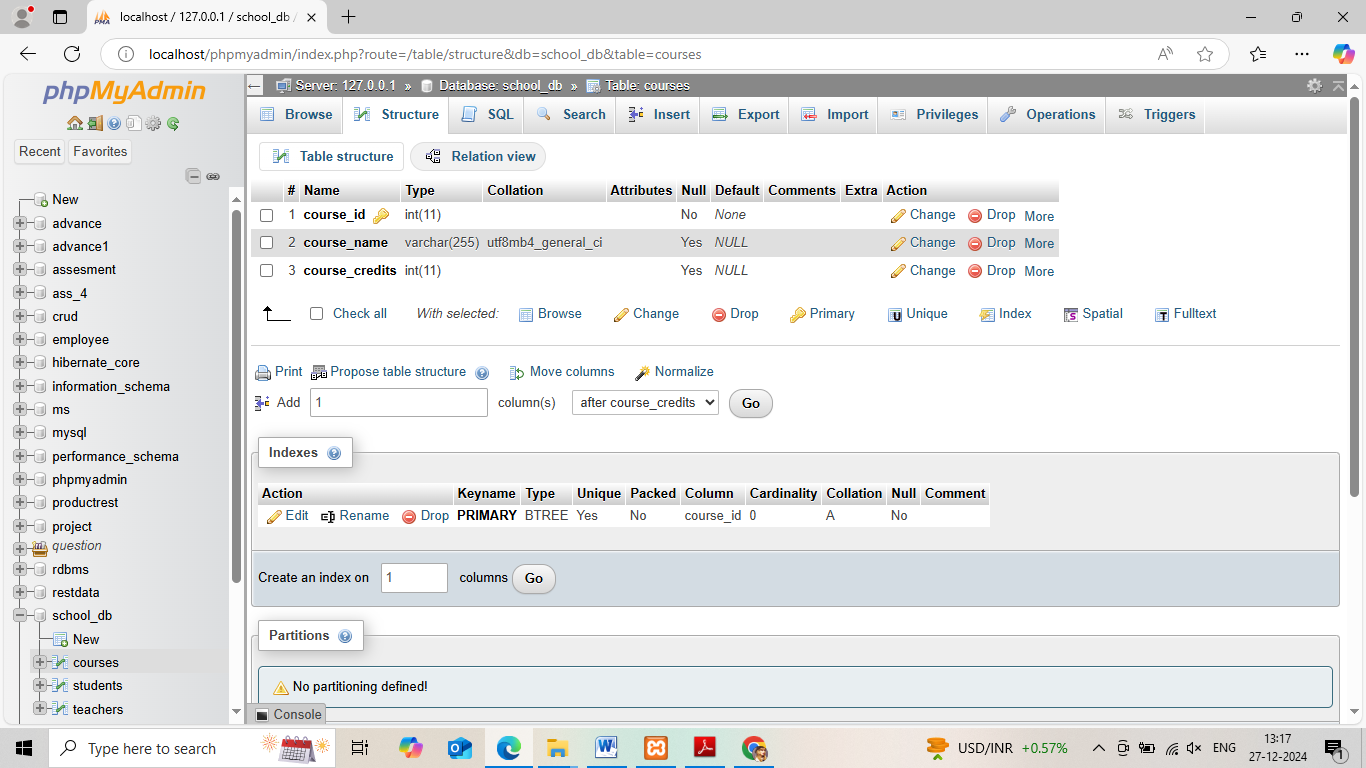
Define the type and range of data that can be stored in a field or column. For example, a field might be defined as an integer, real number, or text.

* **Constraints**

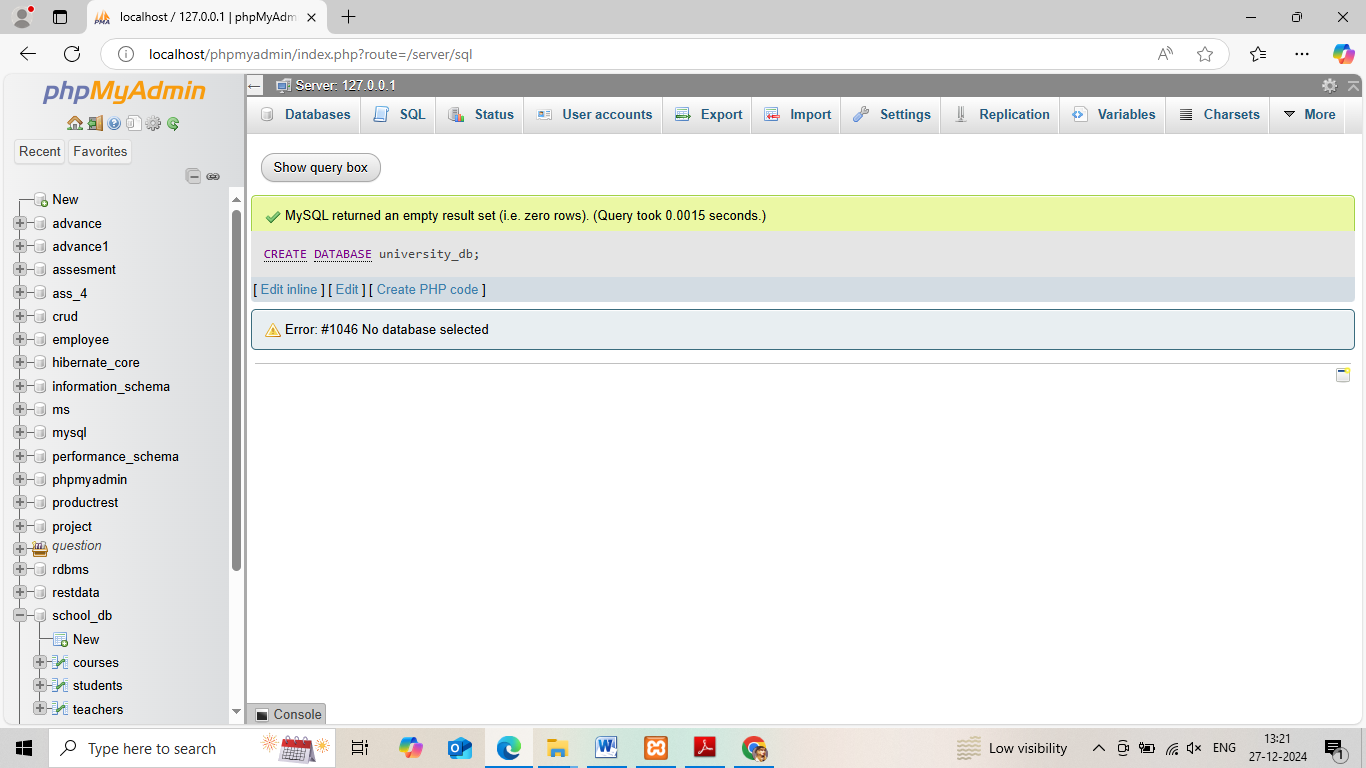
Specify rules for the data in a table, such as a primary key, foreign key, unique, not null, check, or default constraint. If data doesn't meet the constraint rules, the insert operation will be aborted.

**LAB EXERCISES**:

 **Lab 1**: Create a table courses with columns: course\_id, course\_name, and course\_credits. Set the course\_id as the primary key.



 **Lab 2**: Use the CREATE command to create a database university



*5. ALTER Command*

**Theory Questions**:

1. What is the use of the ALTER command in SQL?

* The ALTER TABLE can be used in order to add, drop, delete, or modify the columns in an already existing table. This statement can also be used to add or drop multiple constraints on an already existing table.

2. How can you add, modify, and drop columns from a table using ALTER?

* **Add a column**: Use the ADD COLUMN clause:

ALTER TABLE table\_name ADD column\_name data\_type [1, 4]

* **Drop a column**: Use the DROP COLUMN clause:

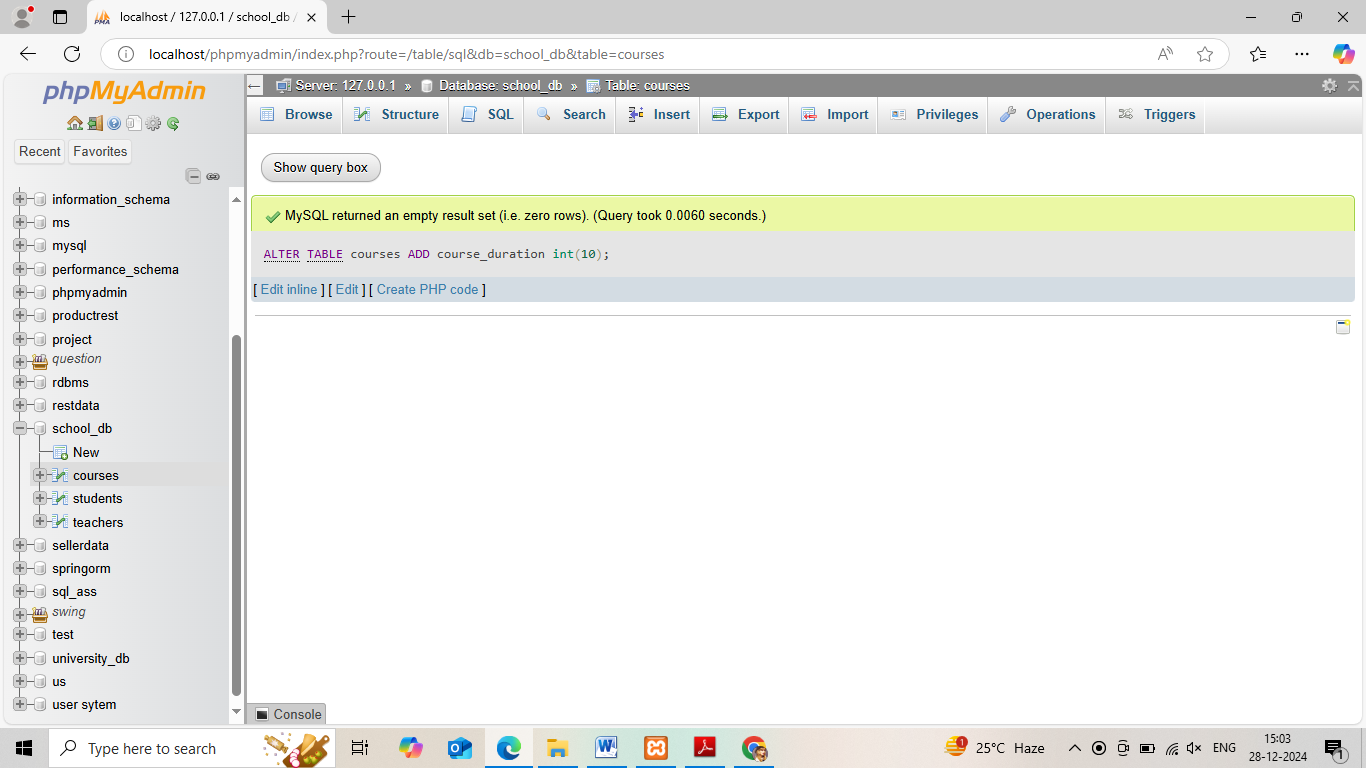
ALTER TABLE table\_name DROP column\_name [1]

* **Modify a column**: Change the column's data type:

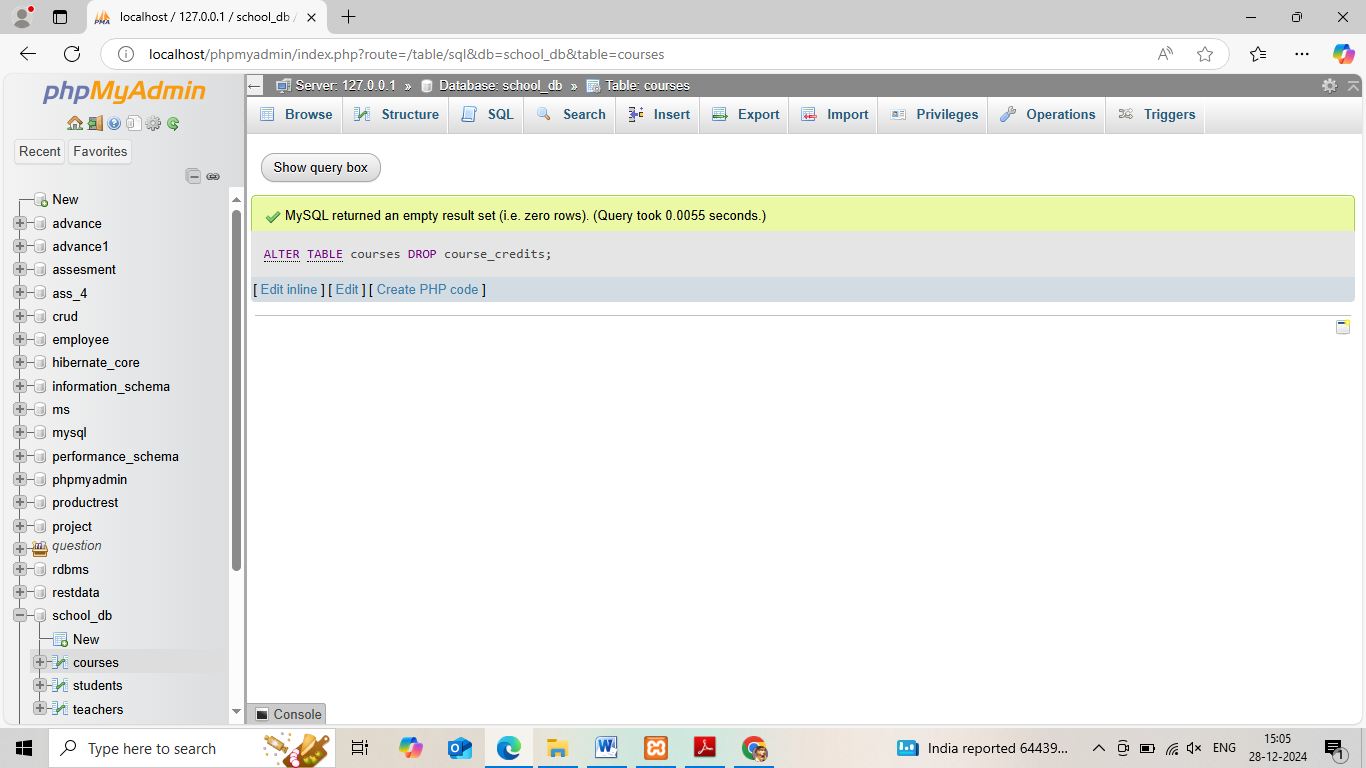
ALTER TABLE table\_name ALTER column column\_name new\_data\_type [2]

**LAB EXERCISES**:

 **Lab 1**: Modify the courses table by adding a column course\_duration using the ALTER command.



 **Lab 2**: Drop the course\_credits column from the courses table.



*6. DROP Command*

**Theory Questions**:

1. What is the function of the DROP command in SQL?

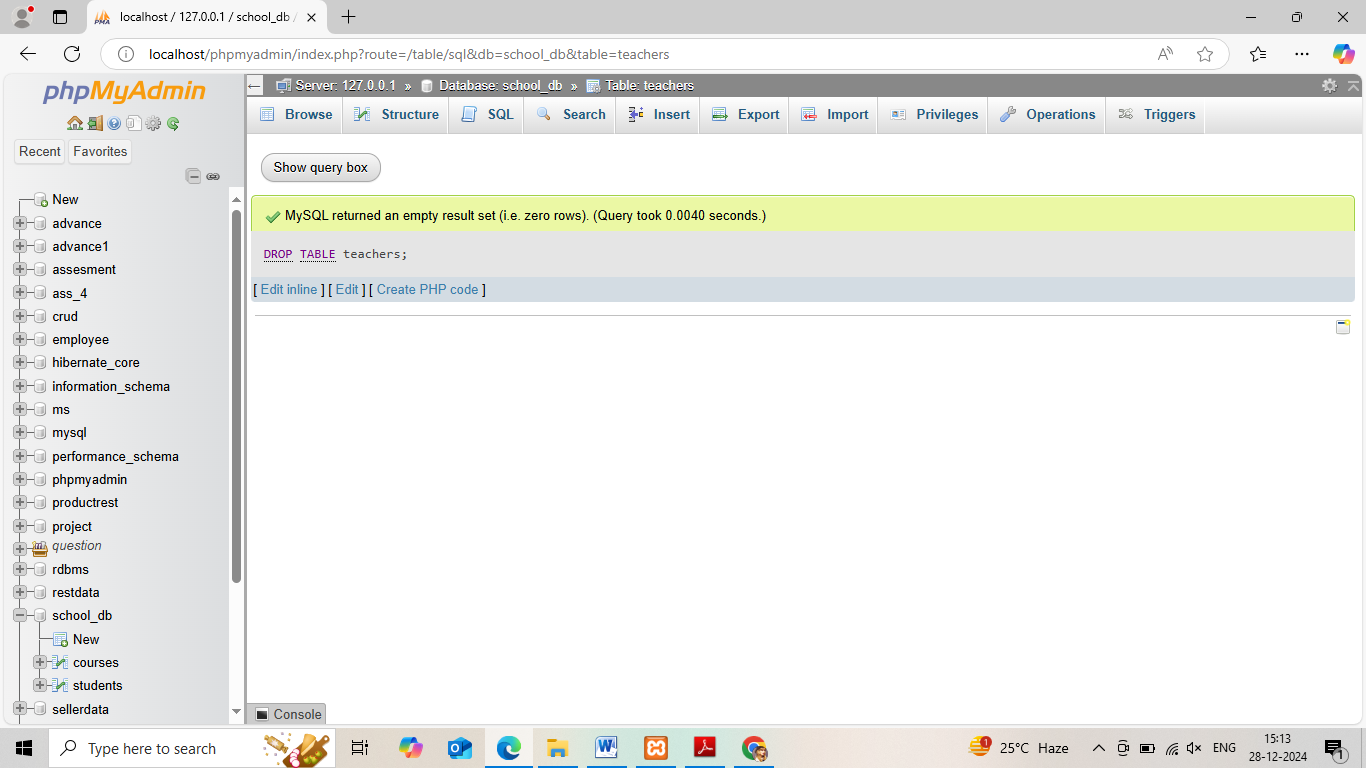
* the DROP command in SQL permanently deletes objects from a database, such as tables, indexes, views, or databases. The DROP command is a type of Data Definition Language (DDL) command.

2. What are the implications of dropping a table from a database?

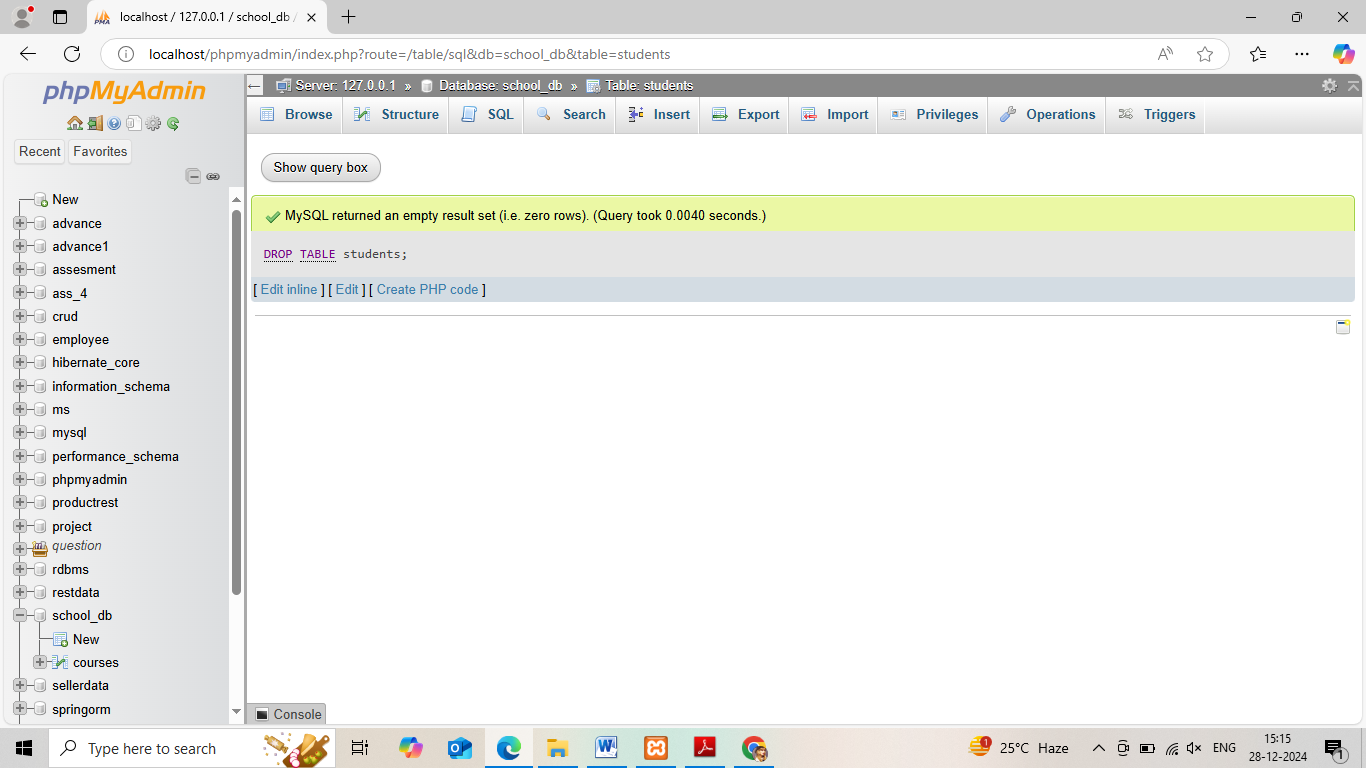
* Dropping a table removes the table definition from the data dictionary. All rows of the table are no longer accessible. All indexes and triggers associated with a table are dropped. All views and PL/SQL program units dependent on a dropped table remain, yet become invalid (not usable).

**LAB EXERCISES**:

**Lab 1**: Drop the teachers table from the school\_db database.



**Lab 2**: Drop the students table from the school\_db database and verify that the table has been removed.



*7. Data Manipulation Language (DML)*

**Theory Questions**:

1. Define the INSERT, UPDATE, and DELETE commands in SQL.

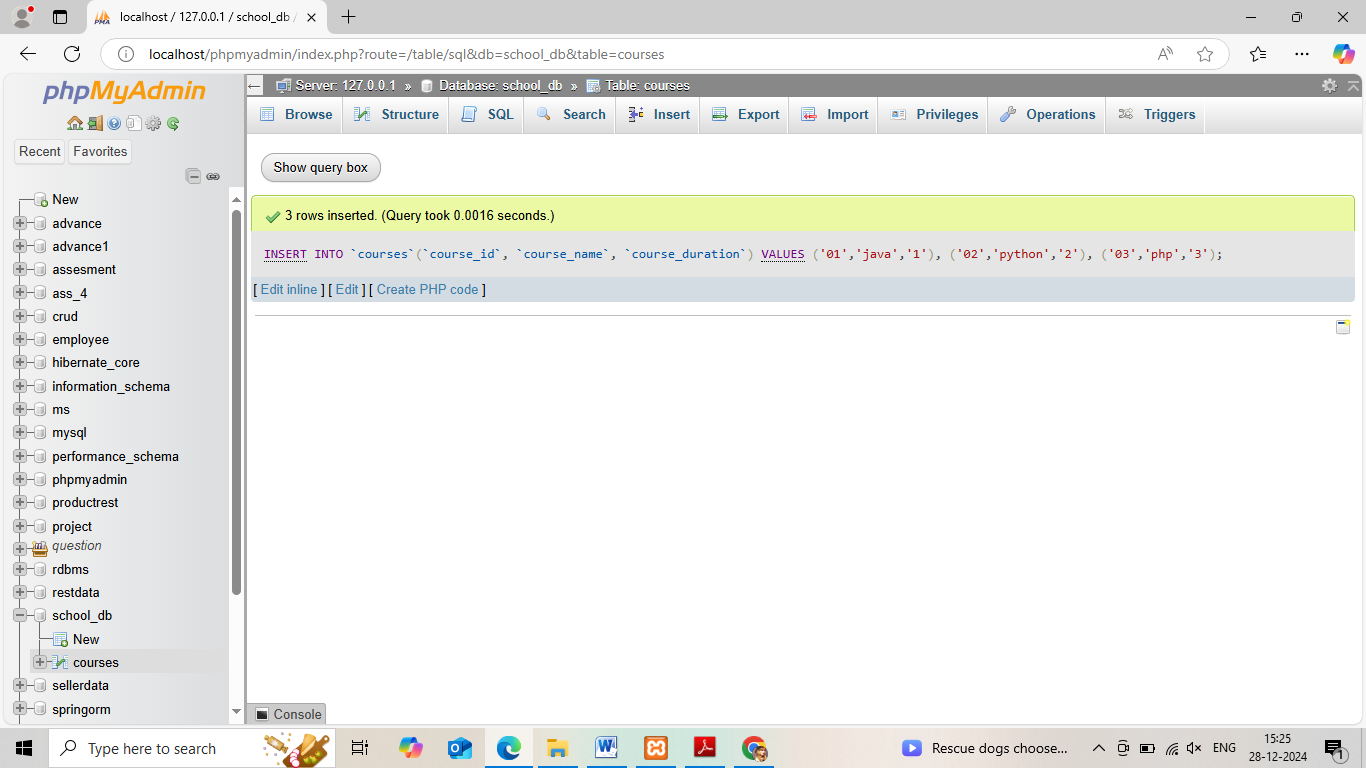
* The INSERT statement allows us to add new data into a table, while the UPDATE statement enables us to modify existing records. On the other hand, the DELETE statement allows us to remove specific records from a table based on specified conditions.

2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

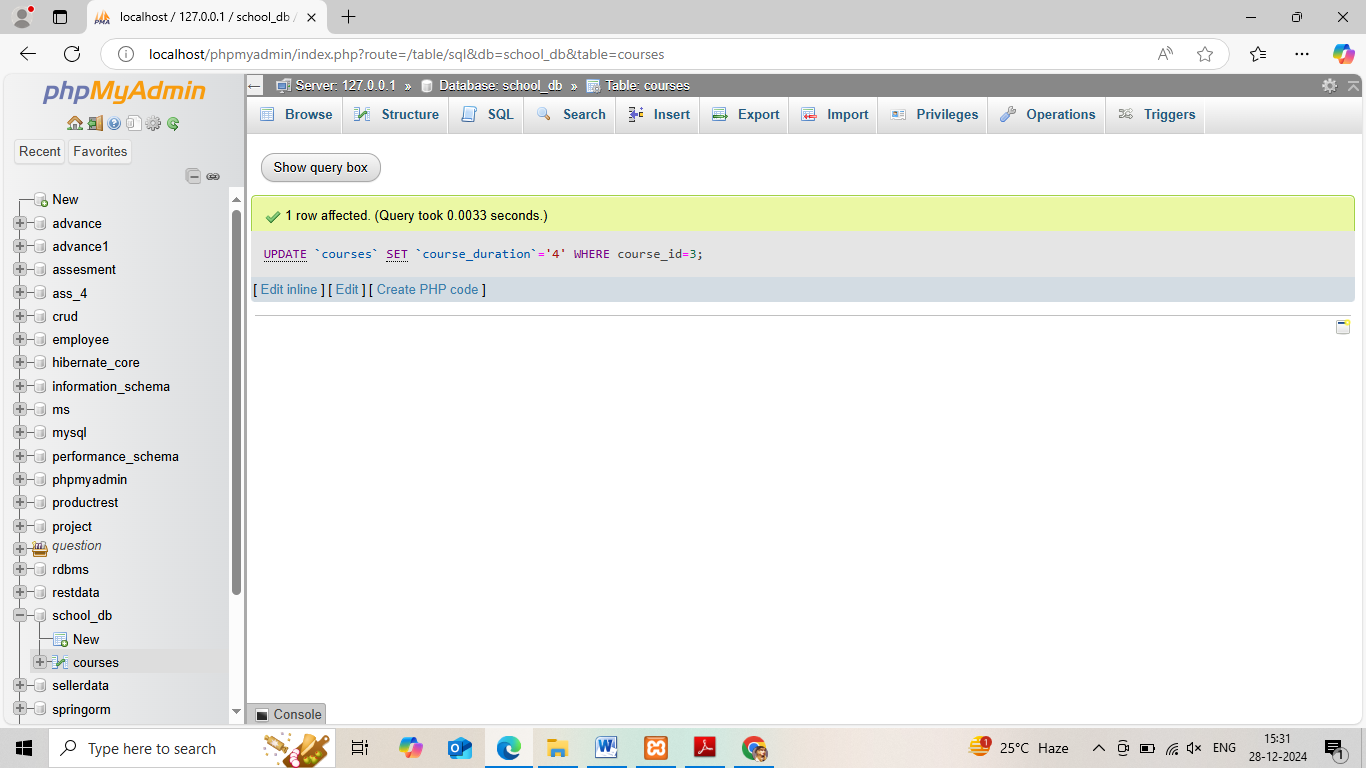
* the WHERE clause is important in UPDATE and DELETE operations because it specifies which records to update or delete, and without it, all records in the table would be affected:
* **UPDATE**: If the WHERE clause is omitted, all records in the table will be updated.
* **DELETE**: If the WHERE clause is omitted, all records in the table will be deleted.

**LAB EXERCISES**:

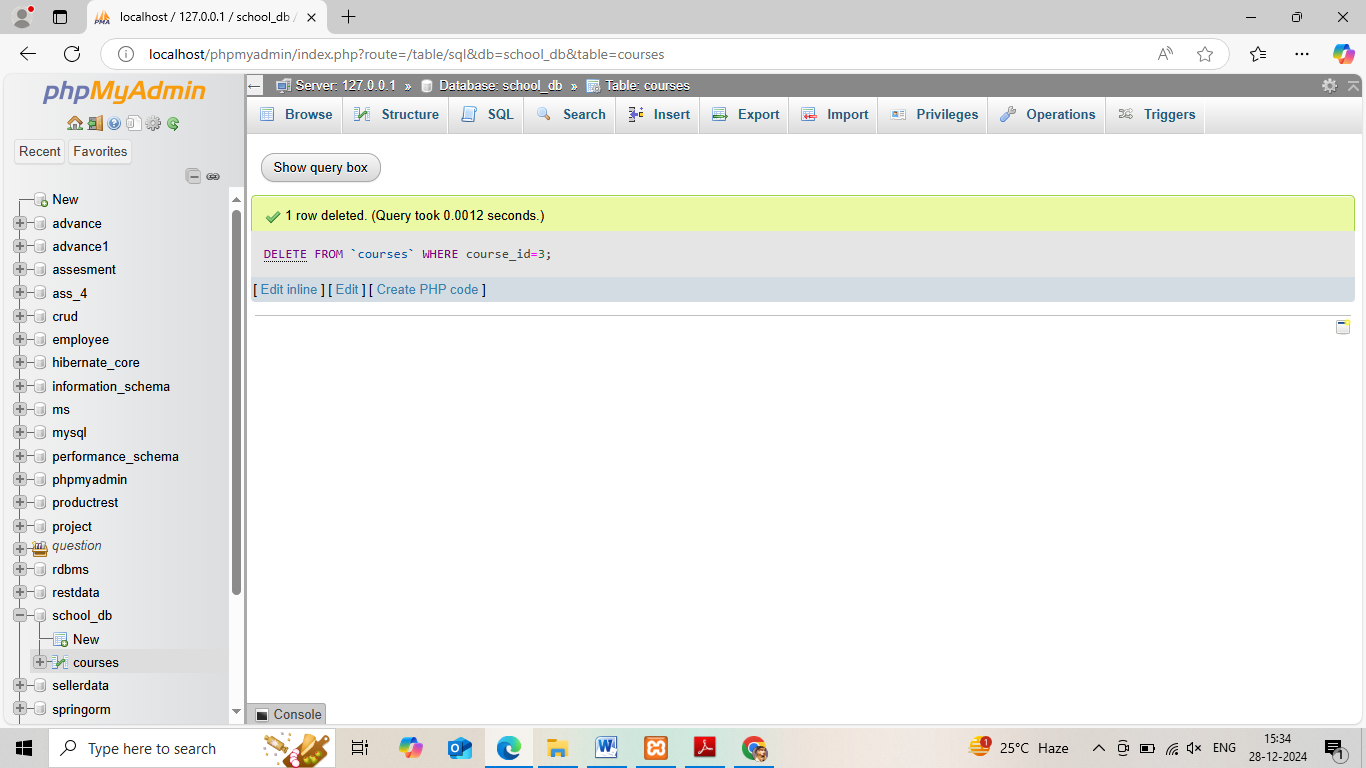
*  **Lab 1**: Insert three records into the courses table using the INSERT command.



**Lab 2**: Update the course duration of a specific course using the UPDATE command.



**Lab 3**: Delete a course with a specific course\_id from the courses table using the DELETE command.



*8. Data Query Language (DQL)*

**Theory Questions**:

1. What is the SELECT statement, and how is it used to query data?

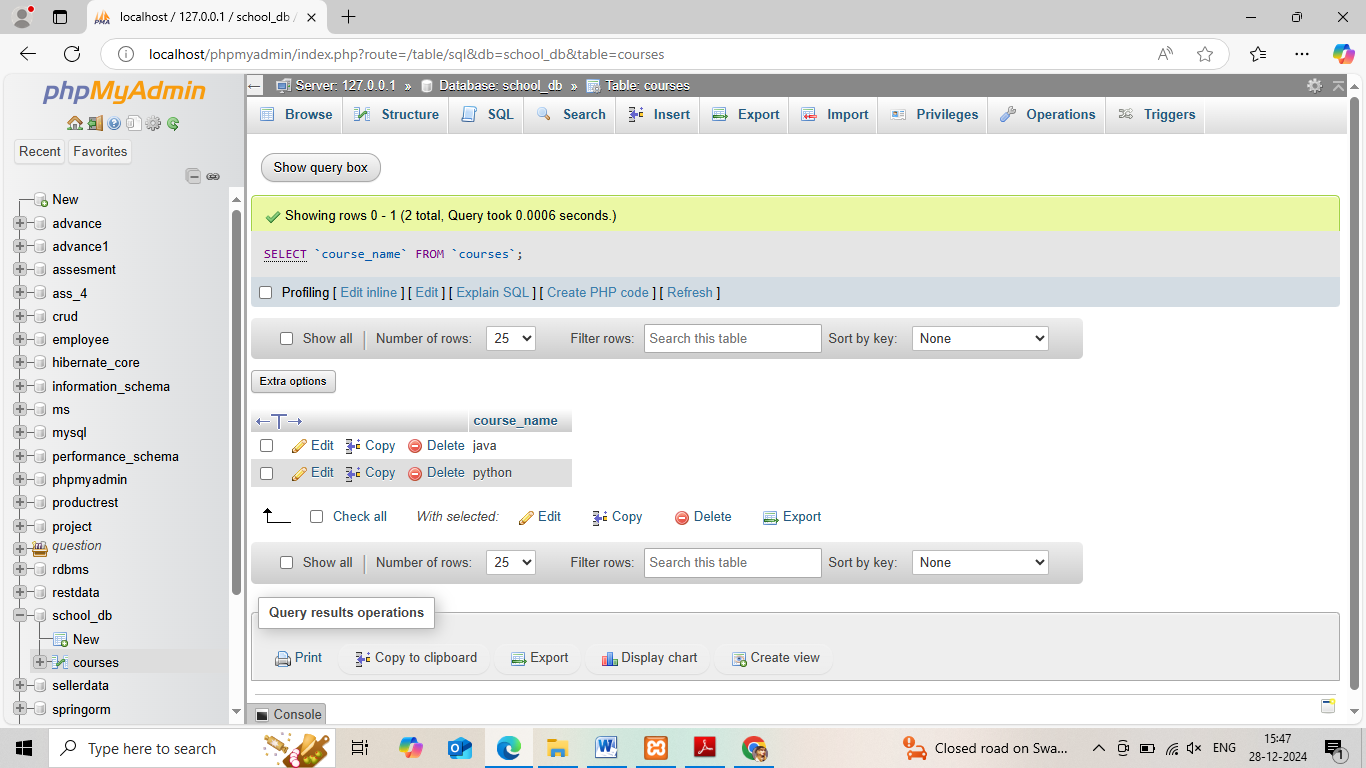
* A SELECT statement retrieves zero or more rows from one or more database tables or database views.
* SELECT is one of the most commonly used SQL statements, allowing you to select specific columns and rows of data from database object(s).

2. Explain the use of the ORDER BY and WHERE clauses in SQL queries.

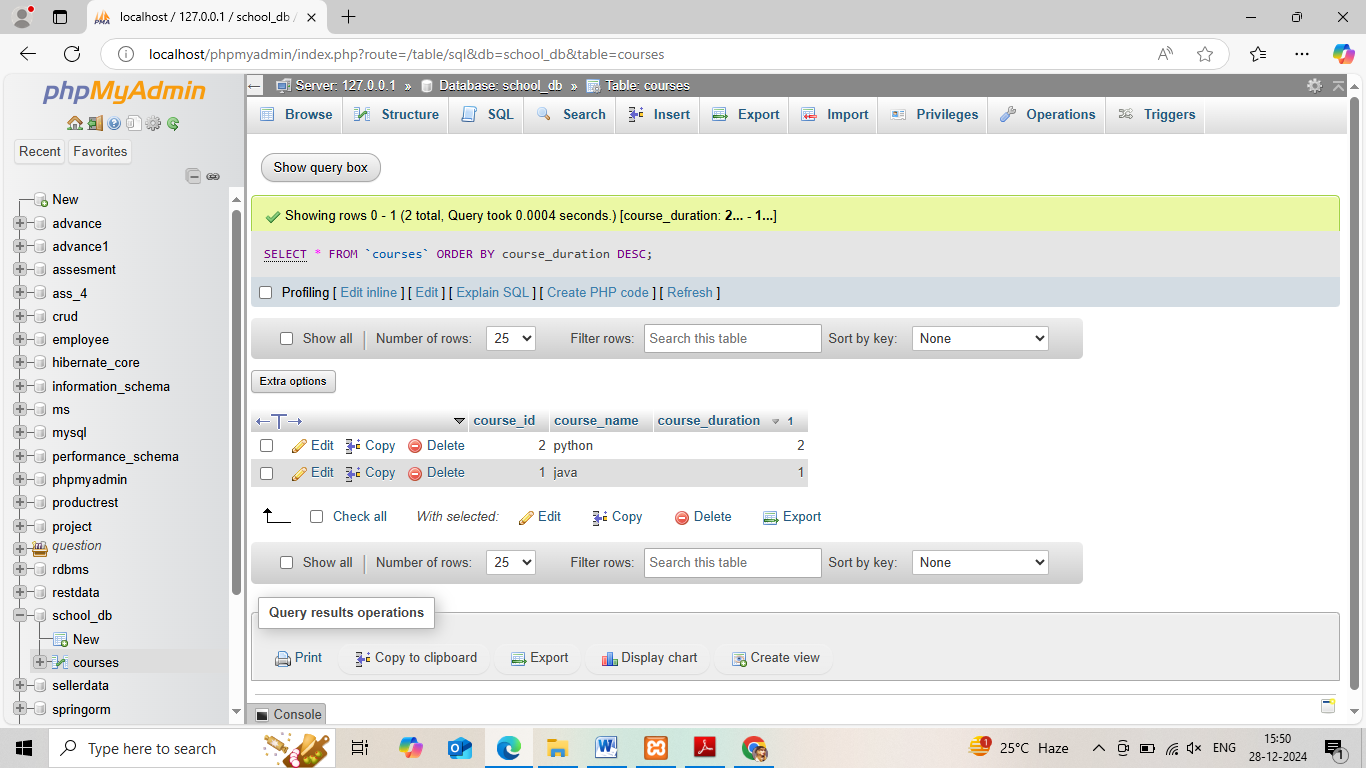
* SQL ORDER BY clause is used after the WHERE clause (i.e. after filtering the data) to sort the result in either Ascending or Descending order. DESC: Keyword is used to arrange the result in Descending Order. Note: Ascending is a default sort order.

**LAB EXERCISES**:

**Lab 1**: Retrieve all courses from the courses table using the SELECT statement.



 **Lab 2**: Sort the courses based on course\_duration in descending order using ORDER BY.



*11. SQL Joins*

**Theory Questions**:

1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFTJOIN, RIGHT JOIN, and FULL OUTER JOIN?

* In SQL, a join is a relational database operation that combines rows from multiple tables based on related columns. The main purpose of a join is to retrieve data from multiple table.
* The main difference between the different types of joins is how they handle unmatched rows:

**INNER JOIN**

The most basic type of join, it returns only records that exist in both tables. INNER is the default join type, so when you write JOIN, the parser actually writes INNER JOIN.

**LEFT JOIN**

Returns all records from the left table, and the matched records from the right table .It includes unmatched rows from the left table.

**RIGHT JOIN**

Works like the opposite of a left join. It returns all records from the right table, and the matched records from the left table.

**FULL OUTER JOIN**

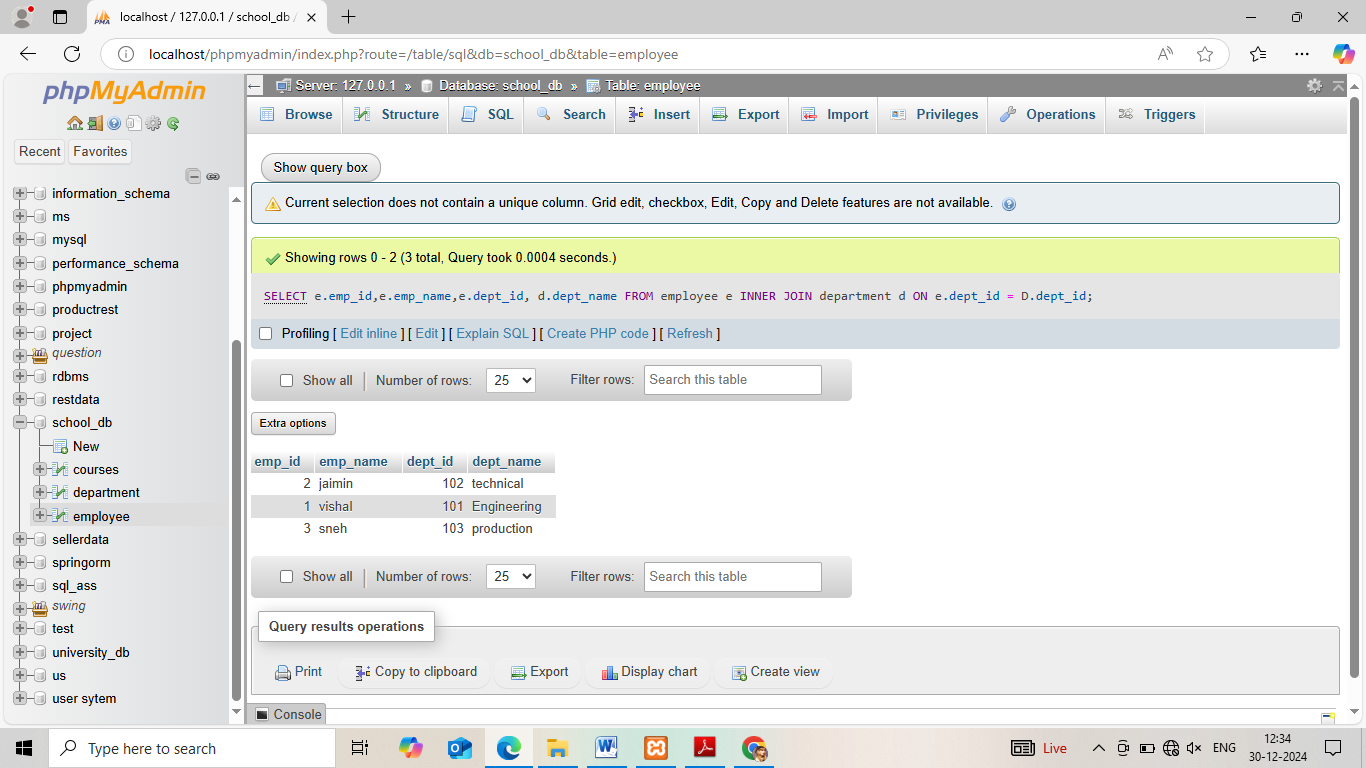
Returns all records when there is a match in either left or right table .It combines a left join and right join.

2. How are joins used to combine data from multiple tables?

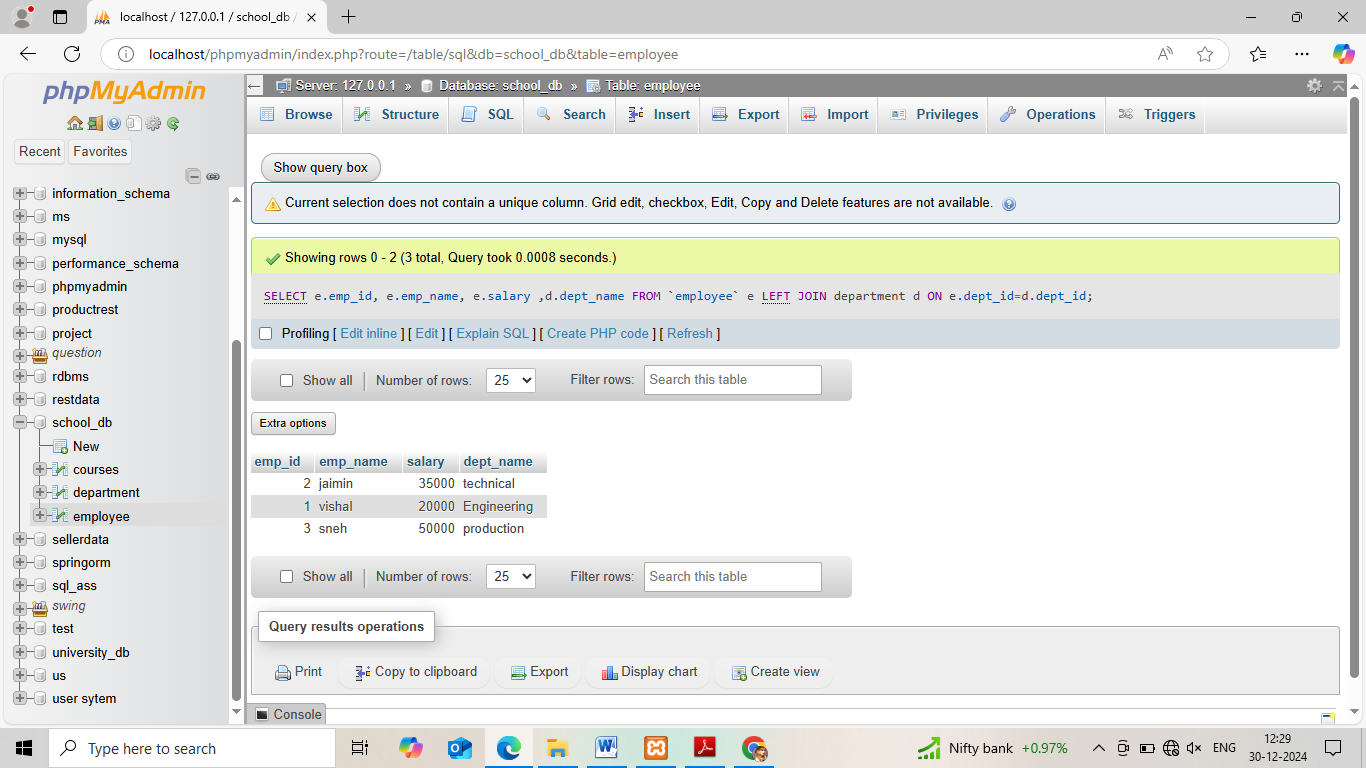
* Using the Inner Join, the tables are combined on the basis of a condition, also known as the join predicate. This condition is applied on the columns of both the tables on either side of the join clause. The query checks all the rows of table1 and table2.

**LAB EXERCISES**:

**Lab 1**: Create two tables: departments and employees. Perform an INNER JOIN to display employees along with their respective departments.



**Lab 2**: Use a LEFT JOIN to show all departments, even those without employees.



*12. SQL Group By*

**Theory Questions**:

1. What is the GROUP BY clause in SQL? How is it used with aggregate functions?

* The GROUP BY clause in SQL groups rows of a table together based on the values in one or more columns, and then applies aggregate functions to each group. This allows you to calculate totals, averages, minimums, and maximums across related records.

Here's how the GROUP BY clause works with aggregate functions:

* **Split**: The rows are split into groups based on the values in the specified columns.
* **Apply**: Aggregate functions are applied to the values in each group.
* **Combine**: The values are combined into a single row.

2. Explain the difference between GROUP BY and ORDER BY.

* **GROUP BY**

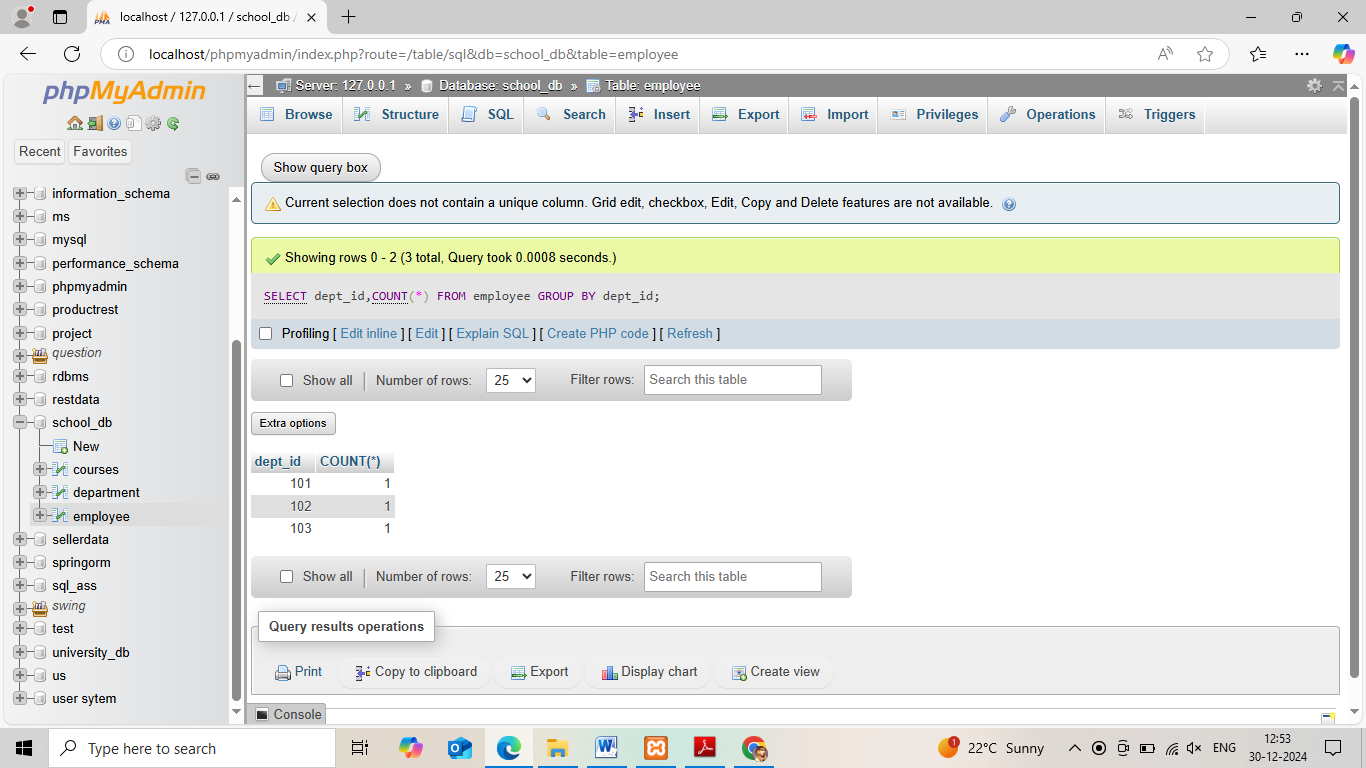
Groups rows with the same values in one or more columns. It uses aggregate functions like COUNT(), AVG(), MIN(), and MAX(). For example, if students with the same subject and year are placed in the same group.

* **ORDER BY**

Sorts the query result by one or more columns in ascending or descending order, alphabetically or numerically

**LAB EXERCISES**:

**Lab 1**: Group employees by department and count the number of employees in each department using GROUP BY.



**Lab 2**: Use the AVG aggregate function to find the average salary of employees in each department.

