**Introduction to Relational Databases and Tables**

In this module, you’ll learn more about relational database concepts and their importance. This module helps you to understand the process of creating a table in your database on MySQL using the graphical interface and SQL scripts. Further, you will also learn how to alter the entries or delete the entries for any table in the database, or even delete the table itself.

**Learning Objectives**

* Describe relational database concepts such as entities, attributes, and primary keys
* Differentiate between DDL and DML statements
* Write commands to create tables in the database
* Explain how to alter, drop and truncate a table

## **Relational Database Concepts**

* **Types of Models**: Understanding different data models, particularly the relational model.
* **Relational Model Advantages**:
  + Provides **data independence**.
  + Uses a simple data structure (tables).
* **Entity-Relationship Model (ER Model)**:
  + Represents entities (tables) and their relationships.
  + Example: In a library, a book can have multiple authors and copies.
* **Entities and Attributes**:
  + **Entities**: Objects that exist independently (e.g., books).
  + **Attributes**: Characteristics of entities (e.g., book title, edition).
* **Mapping to Relational Database**:
  + Entities become tables; attributes become columns.
  + Tables consist of rows and columns.
* **Data Types**:
  + Common types include characters (char, VAR char), numbers (integer, decimal), and timestamps (date, time).
* **Primary Keys**:
  + Uniquely identify each row in a table, preventing data duplication.
* **Foreign Keys**:
  + Link tables by referencing primary keys from other tables.

## **Types of SQL statements (DDL vs DQL)**

* **SQL Statements**: Used for interacting with entities (tables), attributes (columns), and tuples (rows) in relational databases.
* **Categories of SQL Statements**:
  + **Data Definition Language (DDL)**:
    - Used to define, change, or drop database objects (e.g., tables).
    - Common DDL statements:
      * **CREATE**: Creates tables and defines columns.
      * **ALTER**: Modifies tables (e.g., adding/dropping columns).
      * **TRUNCATE**: Deletes data in a table without removing the table itself.
      * **DROP**: Deletes tables.
  + **Data Manipulation Language (DML)**:
    - Used to read and modify data in tables (CRUD operations).
    - Common DML statements:
      * **INSERT**: Adds rows of data to a table.
      * **SELECT**: Reads or selects rows from a table.
      * **UPDATE**: Edits existing rows in a table.
      * **DELETE**: Removes rows of data from a table.
* **Summary**:
  + DDL is for defining/changing database objects.
  + DML is for manipulating data within those objects.

## **CREATE TABLE statement**

**CREATE TABLE Statement**:

* **Purpose**: The video explains how to create a relational database table using the CREATE TABLE statement.
* **Syntax**:
  + Start with **CREATE TABLE** followed by the table name.
  + Enclose the rest of the statement in parentheses.
  + Each column definition includes the column name, datatype, and optional values, separated by commas.
* **Example**:
  + To create a table for provinces in Canada:
  + CREATE TABLE provinces (
  + id CHAR(2) PRIMARY KEY NOT NULL,
  + name VARCHAR(24)

);

* + **Data Types**:
    - **CHAR**: Fixed-length character string (e.g., 2 characters).
    - **VARCHAR**: Variable-length character string (e.g., up to 24 characters).
* **More Complex Example**: Creating an AUTHOR table:
* CREATE TABLE author (
* author\_id CHAR(2) PRIMARY KEY NOT NULL,
* lastname VARCHAR(15) NOT NULL,
* firstname VARCHAR(15) NOT NULL,
* email VARCHAR(40),
* city VARCHAR(15),
* country CHAR(2)

);

* + **Constraints**:
    - **PRIMARY KEY**: Uniquely identifies each row.
    - **NOT NULL**: Ensures certain fields cannot be empty.
* **Key Takeaway**: The CREATE statement is a DDL command used to define entities or tables in a database, specifying attributes, datatypes, and constraints.

## **ALTER, DROP and TRUNCATE Tables**

* **ALTER TABLE Statement**:
  + Used to **add** or **remove** columns, **modify** data types, and manage keys and constraints.
  + Syntax: ALTER TABLE table\_name [action];
  + Example to add a column:

ALTER TABLE author ADD COLUMN telephone\_number BIGINT;

* + Example to modify a column's data type:

ALTER TABLE author MODIFY telephone\_number CHAR(20);

* + Changing data types can cause errors if existing data is incompatible.
* **DROP TABLE Statement**:
  + Deletes an entire table and its data.
  + Syntax: DROP TABLE table\_name;
  + Example:

DROP TABLE author;

* **TRUNCATE TABLE Statement**:
  + Deletes all rows in a table without removing the table itself.
  + Syntax: TRUNCATE TABLE table\_name IMMEDIATE;
  + Example:

TRUNCATE TABLE author IMMEDIATE;

These statements are essential for managing the structure and data within a database.

## **Examples to ALTER and TRUNCATE Tables**

**Reading: Examples to ALTER and TRUNCATE tables using MySQL**

**Estimated time to complete:** 5 minutes

***In the previous video, the ALTER and TRUNCATE syntax applies to DB2. There are variations in syntax between different databases. This reading will explore some examples of ALTER and TRUNCATE statements using MySQL.***

**Objective(s)**

At the end of this reading, you will be able to:

* Use the ALTER TABLE statement in the correct syntax.
* Use TRUNCATE statements in syntax.
* Execute examples of ALTER and TRUNCATE statements.

**ALTER TABLE**

ALTER TABLE statements can be used to add or remove columns from a table, to modify the data type of columns, to add or remove keys, and to add or remove constraints. The syntax of the ALTER TABLE statement is:

**ADD COLUMN syntax**

1. 1
2. 2
3. ALTER TABLE table\_name
4. ADD column\_name data\_type;

Copied!

A variation of the syntax for adding column is:

1. 1
2. 2
3. ALTER TABLE table\_name
4. ADD COLUMN column\_name data\_type;

Copied!

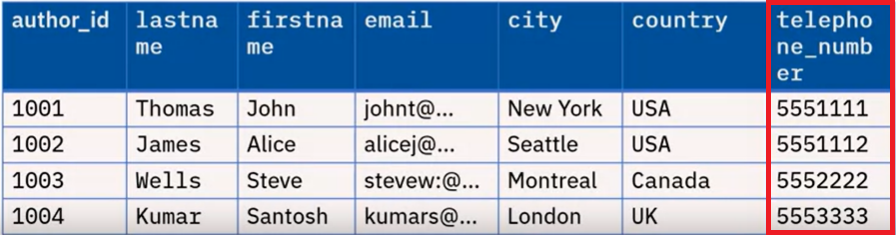
By default, all the entries are initially assigned the value NULL. You can then use UPDATE statements to add the necessary column values.

For example, to add a **telephone\_number** column to the **author** table in the **library** database, the statement will be written as:

1. 1
2. 2
3. ALTER TABLE author
4. ADD telephone\_number BIGINT;

Copied!

Here, BIGINT is a data type for Big Integer.  
After adding the entries to the new column, a sample output is shown below.



**Modify column data type**

1. 1
2. 2
3. ALTER TABLE table\_name
4. MODIFY column\_name data\_type;

Copied!

Sometimes, the data presented may be in a different format than required. In such a case, we need to modify the data\_type of the column. For example, using a **numeric** data type for **telephone\_number** means you cannot include **parentheses**, **plus signs**, or **dashes as part of the number**. For such entries, the appropriate choice of data\_type is CHAR.

To modify the data type, the statement will be written as:

1. 1
2. 2
3. ALTER TABLE author
4. MODIFY telephone\_number CHAR(20);

Copied!

The entries can then be updated using UPDATE statements. An updated version of the "author" table is shown below.



**TRUNCATE Table**

TRUNCATE TABLE statements are used to delete all of the rows in a table. The syntax of the statement is:

1. 1
2. TRUNCATE TABLE table\_name;

Copied!

So, to truncate the "author" table, the statement will be written as:

1. 1
2. TRUNCATE TABLE author;

Copied!

The output would be as shown in the image below.



Note: *The TRUNCATE statement will delete the rows and not the table.*

## **Examples to CREATE and DROP Tables**

**Reading: Examples to CREATE and DROP tables**

**Objective(s)**

At the end of this lab, you will be able to:

* Create and Drop tables in the database.

**Estimated time to complete:** 5 minutes

**CREATE TABLE statement**

In the previous video, we saw the general syntax to create a table:

1. CREATE TABLE TableName (
2. COLUMN1 datatype,
3. COLUMN2 datatype,
4. COLUMN3 datatype,
5. ...
6. );

Copied!

Consider the following examples:

1. Create a TEST table with two columns - ID of type integer and NAME of type varchar. For this, we use the following SQL statement.
2. CREATE TABLE TEST (
3. ID int,
4. NAME varchar(30)
5. );

Copied!

1. Create a COUNTRY table with an integer ID column, a two-letter country code column, and a variable length country name column. For this, we may use the following SQL statement.
2. CREATE TABLE COUNTRY (
3. ID int,
4. CCODE char(2),
5. Name varchar(60)
6. );

Copied!

1. In the example above, make ID a primary key. Then, the statement will be modified as shown below.
2. CREATE TABLE COUNTRY (
3. ID int NOT NULL,
4. CCODE char(2),
5. Name varchar(60)
6. PRIMARY KEY (ID)
7. );

Copied!

In the above example, the ID column has the **NOT NULL** constraint added after the datatype, meaning that it cannot contain a NULL or an empty value. This is added since the database does not allow Primary Keys to have NULL values.

**DROP TABLE**

If the table you are trying to create already exists in the database, you will get an error indicating **table XXX.YYY already exists**. To circumvent this error, create a table with a different name or first DROP the existing table. It is common to issue a DROP before doing a CREATE in test and development scenarios.

The syntax to drop a table is:

1. 1
2. DROP TABLE TableName;

Copied!

For example, consider that you wish to drop the contents of the table COUNTRY if a table exists in the dataset with the same name. In such a case, the code for the last example becomes

1. DROP TABLE COUNTRY;
2. CREATE TABLE COUNTRY (
3. ID int NOT NULL,
4. CCODE char(2),
5. Name varchar(60)
6. PRIMARY KEY (ID)
7. );

Copied!

WARNING: Before dropping a table, ensure it doesn't contain important data that can't be recovered easily.

Note that if the table does not exist and you try to drop it, you will see an error like **XXX.YYY is an undefined name**. You can ignore this error if the subsequent CREATE statement is executed successfully.

In a hands-on lab later in this module, you will practice creating tables and other SQL statements.

## **SQL Scripts - Uses and Applications**

**SQL Scripts**

SQL scripts are a series of commands or a program that will be executed on an SQL server.

SQL scripts are useful for making complex database changes and can be used to create, modify, or delete database objects such as tables, views, stored procedures, and functions.

**Applications of SQL Scripts**

Here are some of the things that you can do with SQL scripts:

* **Create tables**  
  You can use SQL scripts to create new tables in your database. This is useful when you need to add new functionality to your application or when you want to store new types of data.
* **Drop tables**  
  SQL scripts often have commands to Drop tables from databases. This is especially important before Create table commands to make sure that a table with the same name doesnt exist in the database already.
* **Insert data**  
  SQL scripts can also be used to insert data into your tables. This is useful when you need to populate your database with test data or when you want to import data from an external source.
* **Update data**  
  You can use SQL scripts to update existing data in your tables. This is useful when you need to correct errors or update records based on changing business requirements.
* **Delete data**  
  SQL scripts can also be used to delete data from your tables. This is useful when you need to remove old or obsolete records from your database.
* **Create views**  
  Views are virtual tables that allow you to query data from multiple tables as if they were a single table. You can use SQL scripts to create views that simplify complex queries and make it easier to work with your data.
* **Create stored procedures**  
  Stored procedures are precompiled SQL statements that can be executed on demand. You can use SQL scripts to create stored procedures that encapsulate complex business logic and make it easier to manage your database.
* **Create triggers**  
  Triggers are special types of stored procedures that are automatically executed in response to certain events, such as an insert, update, or delete operation. You can use SQL scripts to create triggers that enforce business rules and maintain data integrity.

**Example: Creating Tables**

Let us execute a script containing the CREATE TABLE commands for all the tables in a given dataset, rather than create each table manually by typing the DDL commands in the SQL editor.

Note the following points about these scripts.

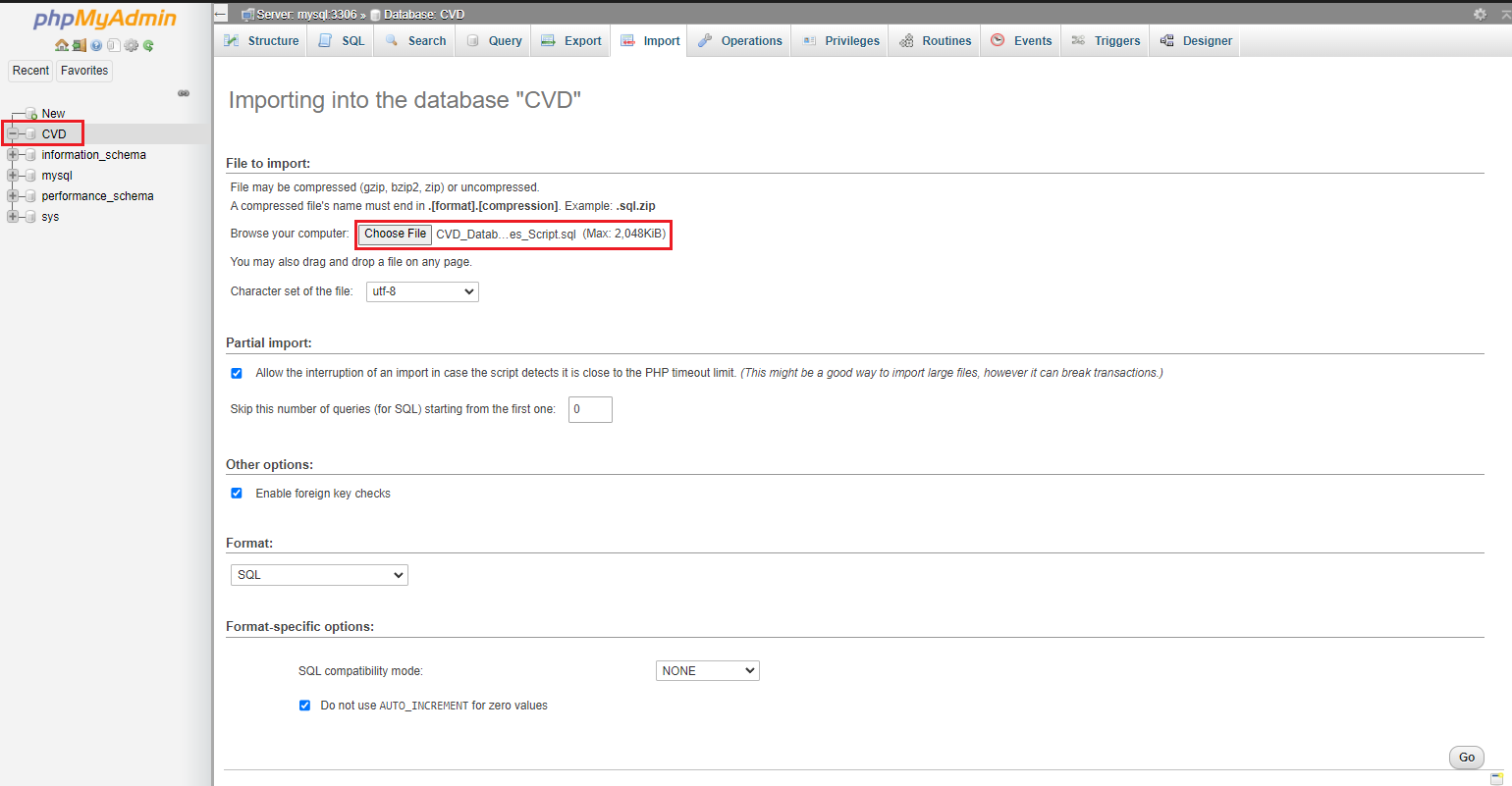
1. SQL scripts are basically a set of SQL commands compiled in a single file.
2. Each command must be terminated with a delimiter or terminator. Most often, the default delimiter is a semicolon ;.
3. It is advisable to keep the extension of the file as .sql.
4. Upon importing this file in the phpMyAdmin interface, the commands in the file are run sequentially.

Consider the following script

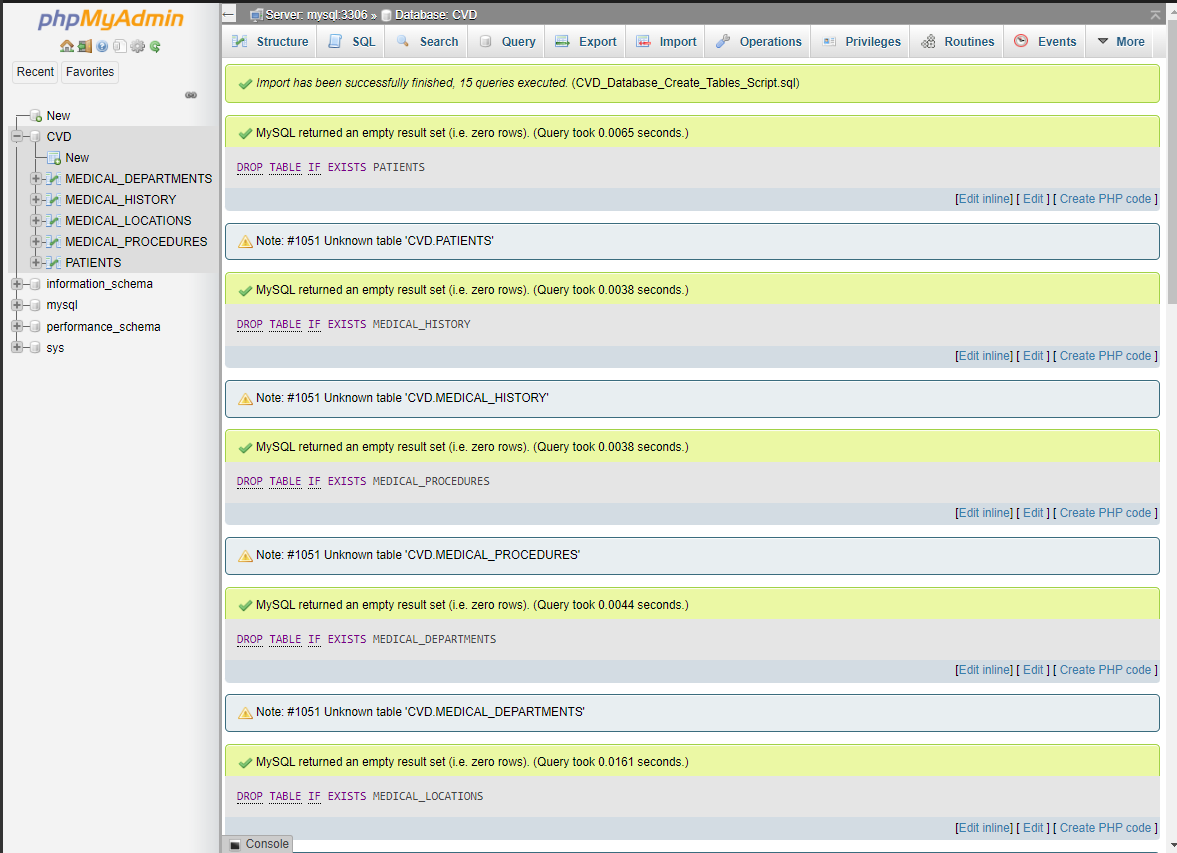
1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. DROP TABLE IF EXISTS PATIENTS;
54. DROP TABLE IF EXISTS MEDICAL\_HISTORY;
55. DROP TABLE IF EXISTS MEDICAL\_PROCEDURES;
56. DROP TABLE IF EXISTS MEDICAL\_DEPARTMENTS;
57. DROP TABLE IF EXISTS MEDICAL\_LOCATIONS;
58. CREATE TABLE PATIENTS (
59. PATIENT\_ID CHAR(9) NOT NULL,
60. FIRST\_NAME VARCHAR(15) NOT NULL,
61. LAST\_NAME VARCHAR(15) NOT NULL,
62. SSN CHAR(9),
63. BIRTH\_DATE DATE,
64. SEX CHAR,
65. ADDRESS VARCHAR(30),
66. DEPT\_ID CHAR(9) NOT NULL,
67. PRIMARY KEY (PATIENT\_ID)
68. );
69. CREATE TABLE MEDICAL\_HISTORY (
70. MEDICAL\_HISTORY\_ID CHAR(9) NOT NULL,
71. PATIENT\_ID CHAR(9) NOT NULL,
72. DIAGNOSIS\_DATE DATE,
73. DIAGNOSIS\_CODE VARCHAR(10),
74. MEDICAL\_CONDITION VARCHAR(100),
75. DEPT\_ID CHAR(9),
76. PRIMARY KEY (MEDICAL\_HISTORY\_ID)
77. );
78. CREATE TABLE MEDICAL\_PROCEDURES (
79. PROCEDURE\_ID CHAR(9) NOT NULL,
80. PROCEDURE\_NAME VARCHAR(30),
81. PROCEDURE\_DATE DATE,
82. PATIENT\_ID CHAR(9) NOT NULL,
83. DEPT\_ID CHAR(9),
84. PRIMARY KEY (PROCEDURE\_ID)
85. );
86. CREATE TABLE MEDICAL\_DEPARTMENTS (
87. DEPT\_ID CHAR(9) NOT NULL,
88. DEPT\_NAME VARCHAR(15),
89. MANAGER\_ID CHAR(9),
90. LOCATION\_ID CHAR(9),
91. PRIMARY KEY (DEPT\_ID)
92. );
93. CREATE TABLE MEDICAL\_LOCATIONS (
94. LOCATION\_ID CHAR(9) NOT NULL,
95. DEPT\_ID CHAR(9) NOT NULL,
96. LOCATION\_NAME VARCHAR(50),
97. PRIMARY KEY (LOCATION\_ID, DEPT\_ID)
98. );

Copied!

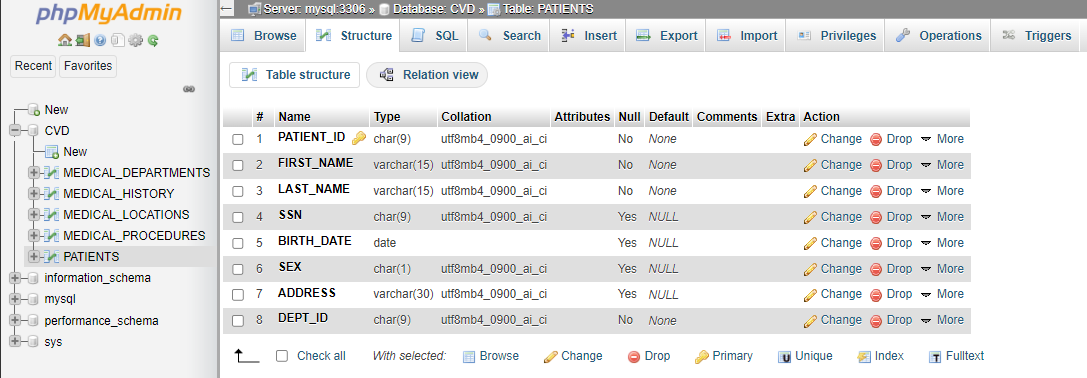
This script incorporates commands to first drop any tables with the mentioned names in the database. After that, the script contains commands to create 5 different tables. All these commands are executed sequentially on the interface.

The contents of this file can be saved in a .sql file format and executed on the phpMyAdmin interface. This can be done by first selecting the database, uploading the SQL script in the provided space, and executing it, as shown in the image below.  


Upon successful execution of each statement in sequence, an note appears on the interface as shown in the image below. It is also prudent to note that the tables created are now visible in the tree structure on the left under the selected database.



You may click any of the tables to see its Table Definition (its list of columns, data types, and so on). The image below displays the structure of the table PATIENTS.

**Summary: Relational Database Concepts and Tables**

Congratulations! You have completed this lesson. At this point in the course, you know:

* A database is a repository of data that provides functionality for adding, modifying, and querying the data.
* SQL is a language used to query or retrieve data from a relational database.
* The Relational Model is the most used data model for databases because it allows for data independence.
* The primary key of a relational table uniquely identifies each tuple or row, preventing duplication of data and providing a way of defining relationships between tables.
* SQL statements fall into two different categories: Data Definition Language (DDL) statements and Data Manipulation Language (DML) statements.